DEPARTMENT OF COMPUTER SCIENCE

Adapting Learning Classifier Systems to Mine Association Rules

Sam Taylor

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Master of Science in the Faculty of Engineering

September 2013 | CSMSC-13

Declaration

This dissertation is submitted to the University of Bristol in accordance with the requirements of the degree of Master of Science in the Faculty of Engineering. It has not been submitted for any other degree or diploma of any examining body. Except where specifically acknowledged, it is all the work of the Author.

Sam Taylor, September 2013

Contents

Abstract

This project has two main areas of study; Learning Classifier Systems (LCS) and Association Rule mining. The output of this project is the ability to mine association rules using a form of LCS; this has given the ability to mine association rules from datasets with both categorical and quantitative or continuous attributes. Both areas of study presented substantial questions to be answered, recently there have been problems identified in LCS causing fit overlapping rules to be competed out the rule population, hence not being presented to the user. The ability to mine association rules using an LCS inspired algorithm was heavily critiqued for its evaluation, of its rule output, therefore the output of this project had to be evaluated to determine its ability of mining interesting and fit association rules. The association rule miner presented in this project is implemented in Java and named XCSAR (eXtended Classifier System for Association Rules), due to many of the mechanisms being inspired by XCS (eXtended Classifier System) the most influential LCS.

This project analysed the implemented algorithm components in an attempt to find the cause of the overlapping issue. This problem was found to be attributed to 3 mechanisms within the algorithm, all of which contributed substantially to fit rules not being presented to the user. Solutions were found to this problem, which were shown to rectify the issue and therefore providing an algorithm which provided rules no matter whether they overlapped or not. This analysis and the provided solutions offer substantial contributions to not only an LCS algorithms ability to mine association rules, but also to the greater LCS community.

Secondly, the implemented algorithm was evaluated. This was carried out using a comparison against well founded association rule miners, such as Apriori and QuantMiner. Additionally, a novel evaluation method was used to determine the interestingness of quantitative association rules; this involved comparing k-means clustering and using plots to visualise clusters, in expectation that an association rule should describe these clusters. It is concluded that the implemented algorithm not only performed very well against other methods and described clusters, but the improvements made upon previous algorithms such as; mining multiple clusters and automatic setting of the maxInterval parameter, created a more flexible and usable algorithm, therefore contributing to the field of association rule mining.

This projects main contributions include implementation, investigation and theoretical therefore all types of project, these main contributions are listed below:

- Determining the main causes of the overlapping competition along with providing and evaluating improvements to the algorithm. This analysis and presented solutions are in Chapter 4.
- Development of an algorithm that has the ability to mine both categorical and quantitative rules online, with limited need for user settable parameters. This algorithm is presented in Chapter 3
- Evaluation of the rule miner against traditional methods as well as using clustering to prove the interestingness and that the rules produced are meaningful, which show that the developed algorithm is able to mine association rules very effectively. This is provided in Chapter 5.

Chapter 1

Introduction

Association Rule mining has the ability to provide rules that describe relationships within a dataset, historically they have been used to aid decisions, in particular, determining what products are bought together in supermarkets and therefore product placement decisions. However there are many other uses for association rules, for example using the detailed description of the dataset, as an initial data exploration technique in data mining. However due to the varied and huge amounts of data now presented to machine learning and data mining algorithms, which is depicted by IBM's 4 Vs of big data seen in IBM (2013). There is a need to improve upon traditional association rule mining techniques to keep up with this ever changing environment. In particular this project will attempt to develop an alternative to traditional methods, enhancing the ability to mine datasets with a variety of attribute types such as categorical and continuous as well as the ability to mine streams of data.

Traditional association rule mining algorithms such as Apriori use an exhaustive approach that was originally only developed to mine binary attributes, however this is very limiting and an exhaustive approach often becomes infeasible for continuous attributes. Furthermore Apriori cannot mine streams of data, due to the need of a full traversal of the dataset to calculate support and confidence values. Therefore to tackle these areas and improve upon previous algorithms, this project will attempt to satisfy this demand by adapting a form of Learning Classifer System (LCS), not only tackling the types of data, but also giving the ability to mine from streams of data online. LCS algorithms are particularly well suited to this due to the use of a genetic algorithm, which have been proven to have the ability to search large parameter spaces such as the combinatorial search space of association rule mining, which grows exponentially in size through the addition of attributes and more so with the addition of continuous attributes.

However there have been problems identified in LCS type algorithms, within the implementation and theoretical content of the algorithm. This problem has recently been identified where when rules describe similar sections of a dataset and therefore overlap, they will heavily compete with each other causing some rules to become extinct and not be presented to the user. This behaviour is undesirable and therefore needs to be analysed to understand the components of the algorithm contributing to this problem. Further to this analysis, new or revised mechanisms are presented to aid the affect of this overlapping competition. Leading from this an algorithm named XCSAR (eXtended Classifier Systems for Association Rules) due to its similarities to XCS (eXtended Classifier Systems). XCSAR is developed in Java and will be the output of this project, of which will be evaluated using a multitude of datasets and against other association rule mining algorithms.

An initial starting point for the implementation of this algorithm, was taken from the implementation of XCSF a function approximator presented in Patrick O. Stalph (2003), whilst some of the mechanisms were used from their source code many of the core elements of the algorithm were removed and rewritten due to the huge contrast between mining association rules and function approximation. However the source code was very useful in terms of a template for the complexity that

is Learning Classifier Systems.

1.1 Aims and Objectives

From the presented motivations for this project above, there are a variety of objectives in terms of; implementation, investigation and theoritical aspects of both association rules mining in general and more specifically LCS algorithms. Therefore to determine the success of this project, the following objectives need to be satisfied:

1. Develop an association rule mining algorithm based on an LCS inspired algorithm, that has the ability to mine both categorical and quantitative attribute
2. Investigate the overlapping competition issue found in recent research, concluding the elements of the algorithm that have a high impact on this problem.
3. Develop and evaluate the solutions for the overlapping problem, determining if any should be used in the output of this project.
4. Evaluate the effectiveness of this algorithm against real and synthetic datasets, compared to traditional implementations of association rules miners such as Apriori. This evaluation will include the accuracy and amounts of rules as well as the systems run time efficiency.

1.2 Structure of Dissertation

This Dissertation will consist of 5 main chapters outlined below:

1. Background Chapter, this chapter will provide an introduction and review of the two main elements of this project association rules mining and Learning Classifier System. This chapter will also highlight problems found within current research, attempting to justify the need for this research, whilst providing the direction for this research.
2. Overview of Algorithm, the contents of this chapter will document the implementation of the algorithm accompanying this report.
3. Analysis and Improvements, this chapter will provide analysis of many of the problems identified in the background chapter, whilst attempting to provide improvements upon existing algorithms and evaluate these improvements.
4. Algorithm Evaluation, the algorithm implemented in this project will then be evaluated against well established association rule mining algorithms. Additionally an attempt to determine the meaningfulness and interestingness of rules given, is provided through the use of clustering.
5. Finally a conclusion chapter will be presented, detailing and summarising this research output including determining the success of the above aims and objectives.

Chapter 2

Background

2.1 Association Rule Mining

Throughout this section, details of association rules will be explored, including a brief introduction to association rule mining, a formal definition and where association rule mining fits into the Knowledge Discovery process. Following from this there will be a discussion on the measures of interestingness for association rules, highlighting the fitness functions available for the implemented algorithm.

Research into new types of association rules expanding the traditional binary approach will be explored, whilst introducing the algorithms to mine association rules of these difference types of rule, including Genetic Algorithms.

2.2 Introduction

Association rule mining was first introduced by Agrawal et al. (1993), where an algorithm was developed to mine sales transaction data to find significant rules to aid a variety of management tasks, ranging from product placement to the design of marketing campaigns. Many of these tasks along with others, such as recommender systems still make use of association rule mining today, almost 20 years from association rules first inception, although Agrawal et al.'s, belief that these methods would be developed into existing database packages has not been realised. Association rule mining has become a large research topic and significant part of the Knowledge Discovery process, along with other popular methods of machine learning such as classification and clustering.

In contrast to classification as an supervised learning task; learning from examples and classifying unseen examples, association rule mining like clustering is a unsupervised method, where the goal of association rule mining is to give a descriptive output in the form of rule, such as; "90% of customers who bought cereal also bought milk". These depict associations within a dataset and allow for rules to be expressed in a succinct readable format to the user. A more general aim for association rule mining in market data datasets in particular is "A customer purchasing item X is likely to also purchase item Y." (Brin et al., 1997a).

Typically an association rule mining task, involves a large database of transactions often taken from sales data. Data from supermarkets have been of particular interest due to the increased collection of data, through the use of loyalty cards. Along with this supermarket managers are often faced with decisions on where to place particular products or marketing decisions. Association rule mining has been prominent in the supermarket data, due to the increasing amount of transaction data being collected through websites and services such as Amazon and NetFlix, there has been substantial research into applying association rule mining to recommender systems such as (Lin et al., 2000). However as discussed association rules can be used simply to described relations within a dataset, therefore can easily be used in the data exploration stage of the knowledge discovery process.

2.2.1 Formal Introduction

A formal introduction of Association Rule mining is as follows; a transaction database is denoted as a set of N transactions T where each transaction t includes a transaction id tid and a subset of items $I = i_1, i_2 \dots i_m$ referred to as an itemset. This itemset can be viewed in the form of a bit vector $t[k] = 1$ or $t[k] = 0$, denoting whether item i_k from I has been purchased during the transaction or not. The output of the rule mining process are association rules, the form of $X \rightarrow Y$ where X, Y are subsets of the set I. These rules meet the minimum confidence constraint specified by the user. An example can be seen below:

$$Cereal \rightarrow Milk$$

where the itemset on the left is denoted as the antecedent and right the consequent.

The above rule can obviously been seen as an intuitive rule that could be mined from a supermarket database, but it is often not easy to numerically determine whether this rule will be interesting to a user. Therefore in the following section will review metrics for association rules. This will give insights to why alternative methods have been developed, along with how they are calculated and what they are trying to evaluate. An example of the rule mining process using a fictional data set, is provided in Appendix A.

2.3 Association Rule Metrics

The metrics of associations rules are particularly important to discuss, as the fitness function for the genetic algorithm will most likely be a form of the presented metrics.

This section will detail the measures that will be used within this project, however both Geng and Hamilton (2006) and Tan et al. (2004) are both very well presented survey papers for measures in association rule mining. These papers provide analysis of the output of the measures and the rankings they provide, therefore this section will focus on the development of measures and why they were introduced, providing an analysis of each. Additionally multi-objective measures will be discussed, which is not included in other survey papers.

These metrics are not only used when determining whether a rule is interesting or not, but they are often used as a heuristics to reduce the combinatorial explosion of association rule mining. Additionally Tan et al. (2004) argues that choosing the right method is paramount, especially due to the conflicting nature of many of the measures. From their study of 21 objective measures for association rule mining, in many cases the measures presented provided substantially different rankings for patterns found in data. Furthermore Tan et al., provided several key properties on choosing the correct measure for particular applications and datasets. This provided an excellent foundation for the design of the fitness function in this project/

It is important understand, what a measure of interestingness is expressing. Tuzhilin (1995), looks at subjective measures of interestingness and suggests that a pattern is interesting if it is unexpected and/or the user is able to provide some action through the knowledge gained through the pattern, this review will consider this to be the correct definition of interestingness for an association rule.

2.3.1 Support-Confidence Framework

Support introduced by Agrawal et al. (1993), is used whilst mining frequent itemsets as a prerequisite measure before rules are evaluated. This provides a large computational advantage in many algorithms as it allows the exponential search space to be pruned. Support also forms the basis of some of the rules to measure interestingness of association rules, in particular Confidence arguably the most well known measure. Support is the proportion of the transactions in the database that include the itemset, and can be defined as follows:

$$supp(X) = P(X)$$

where a frequent itemset is denoted as an itemset with a support that meets the minimum support constraint specified by the user:

$$supp(X) \geq minsupport$$

An important feature of support, its down-ward closure property meaning that all subsets of a frequent set (defined above) are also frequent. Leading from this supersets of a non-frequent set, can therefore not be frequent, this allows pruning of the search space which is used in many algorithms for optimisations of itemset mining.

For example if the itemset $\{A, B\}$ is not frequent, then any additional items added to this itemset will also not be frequent.

The Support-Confidence Framework was also introduced in Agrawal et al. (1993), confidence was used to determine whether a rule extracted from a frequent itemset met a minimum confidence constraint specified by the user.

$$conf(X \rightarrow Y) = P(X \mid Y) = \frac{supp(X \cup Y)}{supp(X)}$$

extracted rule is valid if:

$$conf(X \rightarrow Y) \geq minconfidence$$

Although the Support-Confidence Framework is by far the most well known and used framework for mining association rules, there have been many problems identified. Brin et al. (1997b), noted that as confidence does not use a value for $supp(Y)$, this can provide flaws in the mining process and give counter-intuitive results. This would be especially true if $supp(X \cup Y)/supp(X) = supp(Y)$, which shows the occurrence of X is unrelated to Y. The following example was presented to highlight this flaw:

If people buy milk 80% of the time in a supermarket and the purchase of milk is completely unrelated to the purchase of smoked salmon, then the confidence of $salmon \rightarrow milk$ is still 80%.

It was noted that as this confidence is high and a rule will be generated. However this would be highly misleading and only resulting due to the high support of milk. Leading from this an even more explicit example would be if item X, was present in all transactions therefore $supp(X) = 1$, all rules in the form $Y \rightarrow X$ would have a confidence of 1.

However due to the widespread and excepted use of support and confidence, this will be used in this project however they will be combined due to the lack of pruning ability when using a GA. This

will be further discussed in the next chapter, whilst introducing the XCSAR algorithm.

2.4 Types of Association Rules

Throughout this background section, there has been an emphasis on binary association rules. Since the introduction of association rules there have been numerous advances in utilising all data available and this is an objective of this project. This can include the quantity or time an item was purchased, using the traditional market basket data as an example. The subsequent complexity and information held in these rules, can further aid the decision makers within a business, providing them increased clarification of the data they have stored. Along with this increased knowledge discovery from data, new algorithms and techniques have been developed to mine these types of rules.

This section will document these advancements in using both quantitative and temporal attributes within a database, along with the methods undertaken to mine interesting association rules using this wealth of data. There will be an introduction of quantitative rules, with subsections related to the algorithms used. To provide a summary of what is meant by each of these rule types and the output that can be produced, Table 2.1 provides a lookup table. Additionally the progression from binary rules, to rules of increased complexity will provide a progression in the development of the subsequent algorithm, therefore of particular interest into what rules can be produced along with the techniques used to overcome challenges such as discretising continuous quantitative, is essential for this project.

Type	Method	Example rule	Notes
Categorical	Apriori	$if\ \ profession = accountant\ and\ sex = female$ $then\ native - country = England$	Categories further expand binary association rules, increasing the search space due to the significantly high amounts of values
Quantitative	Interval	$if\ wage \in [5, 10]\ and\ sex = female$ $then\ age \in [25, 30]$	Intervals are formed for the attributes before the mining process, therefore binary attributes are created depicting whether an attribute is in an interval or not is given
Quantitative	Statistical	$sex = female \rightarrow Wage : mean = \$7.90p/hr$ $(overallmeanwage = \$9.02)$	Rules are produced using the distribution of data, as in the example there is deviation from the mean for the rule found.
Quantitative	Fuzzy	$if\ wage\ is\ low\ and\ sex = male$ $then\ age\ is\ young$	Fuzzy rules give linguistic labels, instead of numeric intervals, these labels are usually set before the mining process
Temporal	NA	$if\ wage\ is\ low\ and\ sex = male$ $then\ age\ is\ young$	Fuzzy rules give linguistic labels, instead of numeric intervals, these labels are usually set before the mining process

Table 2.1: Rule Types

2.4.1 Quantitative Association Rules

Traditional Quantitative Approaches

The notion of mining rules from databases including quantitative attributes was introduced by Srikant and Agrawal (1996), details were given of an algorithm to mine quantitative and categorical rules, where binary attributes were considered a special variation of categorical attributes. It was acknowledged that quantitative attributes and categorical attributes, could be mapped to a higher level of boolean attributes. This required partitioning of the quantitative attributes, into intervals. However this approach led to problems such as determining the correct partition. For example, whilst using the support-confidence framework small intervals would lead to low support values, where as large Intervals led to a loss of data, negatively affecting the confidence measure and its robustness.

To overcome this issue all possible continuous values were considered, therefore limiting the problem faced while mining frequent itemsets with the support. The issue with confidence still existed but the use of more intervals reduced the information loss, due to the itemset mining and association rule mining steps are separate in Agrawal and Srikant (1994) both problems were overcome. However this method introduced new computational problems in the form of execution time, due the number of boolean attributes increasing as more intervals were added, causing a huge increase in rules as identified.

The approach to address these problems was to only consider adjacent intervals or values and allowing the user to provide a max support threshold to determine how many adjacent values to looking into. This reduced execution time and the amount of rules subsequently produced by the algorithm.

After the transformation of the attributes to multiple binary values from quantitative and categorical, the original Apriori algorithm could be used to mine for association rules. Although this approach was shown to work, there have been many attempts to provide solutions on how to correctly discretise numeric attributes along with mining for numeric rules.

Statistical Approach

A new method for mining quantitative association rules was presented by Aumann and Lindell (1999), critiquing the method of intervals due to the possibility of misleading and counter-intuitive results, such as the following example; $height \in [100cm, 150cm] \rightarrow age \in [0, 14]$ giving a confidence value of 70%. The method was to provide association rules based on statistical inference theory, arguing that the best description of an numeric attributes behaviour is its distribution, allowing rules to be mined such as in Table 2.1.

To provide these rules the numeric attributes were not discretised, but attributes of the distribution were used such as mean and variance. This allowed the use of statistical tests to determine, when a rule was interesting as the value deviated from the mean. This approach seems robust in terms of the use of statistical measures, along with the output of rules and the intuition this can bring. Although they do provide interesting inferences from a database, there is potential for lack of understanding as a result of the complicated rules and the use of significance. This was also found when the rules were evaluated by a domain expert in the findings of Aumann and Lindell. Additionally multiple numeric antecedents, were not supported leading to the potential loss of interesting

rules.

Genetic Search Methods

Many alternative approaches to mining frequent itemsets have been introduced, in contrast to the Apriori algorithm (Agrawal and Srikant, 1994). These have often involved genetic algorithms, which have been introduced due to their heuristic nature, in an attempt to overcome the computational aspects of the large search space faced while mining for association rules. Due to the search space size growing exponentially when mining quantitative rules, these methods are particularly well suited. Additionally this advantage provided by genetic algorithms is a motivation for this research.

The problem of assigning correct intervals to numeric attributes has been introduced, in correct meaning interesting, minimal information loss and reduction of computational costs. A new algorithm named, QuantMiner was developed in (Salleb-Aouissi et al., 2007), whereby finding good intervals is given as an optimisation problem using the gain measure as the optimisation criteria, however this algorithm used the first step of Apriori in terms of frequent itemset mining. However this has some disadvantages which will be addressed in this research, QuantMiner was restricted in that multiple rules using the same attribute but different intervals could not be mined. For example; $\{X[1,5] \rightarrow Y[1,5]$ and $\{X[10,15] \rightarrow Y[1,5]$, these two rules describe two clusters which could not be mined using QuantMiner. Additionally due to the algorithm mining using a frequent itemset mining approach, it is not well suited for mining association rules online. Therefore this project has been given the objectives to overcome this challenge and improve upon QuantMiner.

Throughout this section Association Rule mining has been introduced including measures of interestingness and the different types of association rules. In the next section the other part of this project content will be introduced, including the XCS algorithm and using elements introduced in this chapter the current state of association rule mining in the LCS community.

2.5 Learning Classifier Systems

This section will provide an overview of the eXtended Classifier System (XCS), the history of how XCS has been developed through the field of Learning Classifier Systems (LCS). This is particularly relevant to this work as an algorithm to mine association rules is the output of this project, which has been developed based on an implementation of XCS. The main aim of this project is to provide a definite answer, as to whether XCS has the ability to mine association rules. Along with the wealth of other tasks it has undertaken such as; classification, function approximation and multiplexer problems to name a few.

XCS has been said to have provided a pivotal point in the history of LCS research, leading to many of the present LCS algorithms using XCS as a basis or using portions of the XCS framework. This in part is why it is an interesting research topic to fully understand if association rule mining can be added to XCS's repertoire.

Additionally XCS uses both, a genetic algorithm and reinforcement learning, to mine "If $\cdots$ Then" rules to carry out a multitude of tasks. It is therefore logical that; XCS can also be modified to give the ability to mine association rules, in fact this has been proven in Orriols-Puig and Casillas (2010) of which will be reviewed and improved upon in this project.

2.6 Learning Classifier System Overview

Learning classifier systems have evolved through many aspects of computer science, as well as other sciences such as Biology. An excellent survey article, depicting the history of LCS and its advances is presented in (Ryan J. Urbanowic, 2008).

The main components of an LCS system include, the discovery process whereby new rules are discovered and added to the population of classifiers (the If $\cdots$ then rules) and the learning process (or credit assignment more specifically for XCS). The learning component handles, updating of parameters for each classifier, such as their fitness. This has two functions identified in Ryan J. Urbanowic (2008) of identifying classifiers which will be useful for obtaining future rewards and for use of discovering new rewards. These components interact with the systems environment, which in the example of association rules is the database of transactions.

Learning classifier systems were first introduced by Holland (1976), with the first implementation being introduced in (Holland and Reitman, 1977). Due to this being developed at the University of Michigan, similar types of learning classifiers systems which use the same approach of evolving rules individually have been named "Michigan-Style". In contrast to this an approach developed at, the University of Pittsburgh evolved sets of rules, allowing these set of algorithms to be given the umbrella name of "Pittsburgh-style" algorithms. The XCS algorithm reviewed here is a Michigan style algorithm, introduced in (Wilson, 1995). Ryan J. Urbanowic entitles the section introducing XCS, as the "The Revolution" due to the contribution of XCS as a simplified LCS, whilst having the ability to mine general and accurate classifiers. The reason for this in part was attributed to intuition of using accuracy as the fitness function instead of strength, this was thoroughly researched in (Kovacs, 2002).

2.7 XCS Algorithm

To give the ability to explore the different aspects of XCS and potential problems that have been identified, which will affect the implementation of this project, the XCS algorithm will be introduced. This introduction will be in three sections; rule discovery, learning process and the production system as is standard. It is not seen as possible or necessary in this background, to cover the XCS algorithm in depth, due to the complexity. However both Kovacs (2002) and Butz and Wilson (2001), provide a rigorous explanation of the algorithm and its inner workings. Before introducing the XCS algorithm, the representation used in XCS will be introduced.

2.7.1 Rule Representation

The rule representation in LCS systems are often in the form of ternary strings, where a rule is given as $condition \rightarrow action$. The condition can have a set of the following values, $\{0, 1, \#\}$. Where $\#$ is a wild card, meaning either 0 or 1 can match that item. An action can take a value from $\{0, 1\}$, this could provide a potential restriction while mining association rules, as this representation would only allow one consequent. However there have been implementations of XCS that use a multi-step approach, which can multiple actions and therefore multiple consequents.

An interesting trait of XCS is its memory less design, in that XCS only has knowledge of its most

recent example from its environment. This trait is particularly useful in this research due to the ability to implement an online association rule algorithm.

2.7.2 Rule Discovery

XCS uses a fixed population of rules or classifiers, in the form previously presented. This is often expanded to the use of micro-classifiers, which includes a numerosity value determining the number of times the rule was discovered, this aids the run time of algorithm by reducing duplicated rules in the population.

Rule discovery deals with generation and deletion of new rules, generation can occur using three methods; *a*) Random initial population *b*) Covering and the *c*) Genetic Algorithm . In the random initial population, rules are created based on probabilities for each of their items. Covering allows new rules to be added that are similar to the current environmental input. This is particularly important when a rule does not have a match set, which will be introduced in a subsequent section. The GA used in XCS is a niche genetic algorithm, which restricts mating to parents that are similar in terms of the instances they cover. This has recently been found to introduce problems in the rule discovery process, which will be of particular importance to this research, as it has the potential effect the association rule mining process.

Deletion is also handled by the GA in the form of a probability to determine whether the rule should be deleted, this probability increases when a rule has matched fewer inputs. Subsumption deletion also occurs, whereby offspring are subsumed by their parents if the parents are highly accurate or the parent is a superset of the offspring. If this happens the numerosity of a parent is increased and the offspring is not added to the population.

2.7.3 Learning Process

This section of the XCS algorithm, often named the credit assignment system updates the parameters given to each rule including prediction (its estimated reward), prediction error and fitness.

2.7.4 Production System

The XCS production system handles the interaction with the environment, in terms of accepting input of a new example, creating a match set from that example and selecting actions to undertake. The match set for an input is created from the population of classifiers, based on the rules that match the current input from the environment. A system prediction is then calculated for the match set, based on the prediction or action of each classifier:

$$P(a_i) = \frac{\sum_{c \in \{M\}_{a_i}} F_c \cdot p_c}{\sum_{c \in \{M\}_{a_i}} F_c}$$

where $\{M\}_{a_i}$ is the the subset of the match set $\{M\}$, which have the same action a_i and p_c is the prediction of rule c, F_c is the weighted average of fitness for the rules within the match set.

This system prediction is then used to to determine which action should be undertaken, if there are no rules in the match set, the covering algorithm is used to guarantee there is at least one action that can be undertaken. The action set $\{A\}$, is then created based upon the action selected. A

reward r_t is then returned from the environment for the selected action. The action set created is used in XCS niche GA, to attempt to find new rules.

The fitness of rules are also updated from the reward given from the environment, this is shared among the rules within an actionset, where a rule gains more fitness when it has higher numerosity.

Now the basics of the XCS algorithm have been introduced,with particular emphasis on research that directly affects this project will be presented. In particular the next section will document recent findings within XCS research into problems that have been identified within the XCS algorithm.

2.8 XCS cannot learn all boolean rules

Whilst applying XCS to a particular subset of tasks that involved evolving binary rules Ioannides et al. (2011b) found that XCS provided counter-intuitive and erroneous results. The results displayed were thought to provide substantial problems when implementing any association rule miner using a XCS based algorithm. Therefore this problem was thought to be present within CSAR Orriols-Puig and Casillas (2010), especially as the problem has been identified more recently than CSAR's implementation.

Within Ioannides et al. (2011b) XCS was used in the field of digital Design Verification in the attempt to learn boolean rules, this is very similar to learning boolean association rules. However one difference which is particularly important, is association rule miners generally output many more rules than a classifier. It was found that rules, competed with each other to such a extent, that slightly less fit rules were deleted from the population. This happened due to the overlapping rules being present in the same match set and the rules with higher fitness being selected more often for reproduction, therefore often increasing that rules numerosity. Rules with slightly less fitness although still fit had lower numerosity and therefore were deleted, these rules were often rediscovered by the GA but deleted once more. The fact that association rule miners output many rules, gives this problem more precedence as the rules with slightly lower fitness could be competed out of the population and not shown to the user.

Inherently association rule mining is a binary problem, even if quantitative attributes are used attributes are generally classed as being in a interval or not. Therefore it was thought that this identified problem, will also have an affect on mining association rules. Further experimentation and proving that this is the case is presented in Chapter 4, this chapter also further expands the definition of overlapping rules whilst providing examples, within the context of association rule mining.

Potential solutions to this problem were introduced in Ioannides et al. (2011a), which included using a Panmictic GA due to the use of a Niche GA causing extreme competition within small subsections of rules. This solution among other is explored in the Chapter 4, which also provides analysis upon the effectiveness of these solutions.

This section on overlapping competition has been short, as much of the explanation and analysis is presented in Chapter 4, which is a main contribution to this research.

2.9 Association Rules in XCS

Mining association rules in XCS has been identified to be feasible due to the nature of the output of XCS, in terms of "If then" rules. There has been one presented adaption of XCS to solve this

problem in Orriols-Puig and Casillas (2010), this algorithm uses a XCS and a variation of XCS called UCS presented in (Bernad-Mansilla and Garrell-Guiu, 2003). UCS is an adaptation of XCS which evolves rules to classify data, using fitness as the portion of examples a classifier covers.

CSar was developed to mine quantitative association rules as introduced in the previous chapter, a similar technique as Mata et al. (2002) and QuantMiner in Salleb-Aouissi et al. (2007), where the intervals for the quantitative attributes were evolved, however unlike QuantMiner CSar also evolves rules rather than carrying out the preprocessing step of finding frequent itemsets. This technique allows CSar to be able to learn from incremental streams of data, along with the memory less design of XCS.

The details of CSar will now be discussed as this will be an important starting point of the subsequent algorithm, additionally the algorithm will have to be refined in light of the research reviewed. Therefore once the algorithm is outlined, the elements thought to cause problems while applying CSar will be explored and analysed in a subsequent chapter. Further experimentation is taken place in Chapter 4 to prove these theoretical assumptions taken, in particular the recent research into overlapping problems identified above. The rules CSar evolves are output in the following form:

$$if\ x_i \in v_i\ and \ldots\ and\ x_j \in v_j\ then\ x_k \in v_k$$

Where the antecedent can have any amount of attributes, denoted by x however the consequent can only hold one attribute or action. Each attribute is given an interval v, which has both an upper and lower bound. Intervals are given a maximum length within CSar to ensure that an interval does not cover all examples and therefore have a high support. Additionally CSar can mine categorical attributes, which can be included in the traditional binary rules.

CSar has three main components which approximately follow the components in the XCS algorithm, these include; the learning process, rule discovery process and rule reduction which will now be discussed in turn. As a prerequisite each classifier, as in XCS is given a set of parameters, which are given as: *a*) Support of the rule *b*) Confidence of rule *c*) fitness of the rule *d*) antecedent support *e*) consequent support *f*) numerosity of the rule and *g*) the creation time of rule.

2.9.1 Learning Process

During the learning process in CSar, an example is fed into the system where a Match set $\{M\}$ is created from the rules whose antecedent matches the example. It is of particular importance to note that, as input are presented as itemsets from the environment and not rules, there could be many rules that match an example. This leads to matched rules having different antecedents and consequents. It can be seen that this process is similar to how frequent itemsets can be split into association rules, as in the Apriori algorithm. However in this case the itemset is just one example from the environment. Additionally as in XCS, if there are not enough rules in the match set a covering operator is executed.

Once the match set has been created an action set or association set is created, this is particularly important to discuss due to the problems highlighted in the previous section, when XCS is presented with a problem where overlapping rules occur within the dataset. Two alternative methods were trialed for creating an association set as follows.

Group by Antecedent

The first method attempted to group the match set by antecedent, where N_a association sets are created and each set includes rules that have the same antecedents. A simple binary example of this process can be created from the match set shown in Table 2.2, from this the association sets would be created as shown in Table 2.3.

Rule No.	Antecedent	Consequent
1	$\{Milk, Cheese\}$	$\{Butter\}$
2	$\{Milk, Cheese\}$	$\{Bread\}$
3	$\{Milk, Cheese, Bread\}$	$\{Butter\}$

Table 2.2: Match set for example $\{Milk, Cheese, Butter, Bread\}$

Association Set	Rules
1	$\{Milk, Cheese\} \rightarrow \{Butter\}$ $\{Milk, Cheese\} \rightarrow \{Bread\}$
2	$\{Milk, Cheese, Bread\} \rightarrow \{Butter\}$

Table 2.3: Association Sets

From association set 1 and noted in Orriols-Puig and Casillas (2010), rules grouped using this strategy can have different consequents. It is believed that these rules would overlap and therefore compete against each other. This would lead to optimal rules, although showing different information having lower fitness. This will need to be reviewed further with an implementation, but the results presented in Ioannides et al. (2011b), suggest that this will affect the mining process significantly.

Group by Consequent

The second method of finding the association sets, is grouping the match set by consequent. This grouping is based on the consequent having an equivalent value, where equivalent means the intervals of the attribute overlapping. This means that the rules in the same association set may have different values in the antecedent and over different intervals, which could also contribute to the overlapping problem identified.

It was found that grouping by consequent provided significantly less rules and for this reason was the favoured technique, therefore XCSAR used this approach which is detailed in the next chapter. Additionally, grouping by antecedent is thought to further exacerbate the overlapping problem.

2.9.2 Rule Discovery

The genetic algorithm element of CSar is triggered when a certain amount of time has passed, this is carried out on an association set. Two parents are selected for reproducing, using the probability as defined below:

$$p_{sel}^{k} = \frac{F_k}{\sum_{i \in [A]} F^i}$$

F, is the fitness of a classifier calculated using support and confidence as follows:

$$F_i = (conf_i \dot{s} upp_i)^v$$

where v is a parameter that puts pressure towards classifiers that are highly fit.

The parents are then subjected to crossover and mutation based on certain probabilities, the new offspring are checked to subsumption and added to the population.

2.9.3 Rule Reduction

After the learning process has been completed, the rule set is reduced to ensure rules have matched enough inputs, rules are also checked for subsumption again. There is also a check to ensure rules in the population are different from each other, this forms the output of the algorithm. It is not clear when this rule reduction was carried out and how this relates to providing an algorithm that is able to mine association rules online and incrementally. This technique is developed and discussed in the next chapter of this project.

This section has briefly outlined the method of which association rules mining has been carried out with a version of XCS, this will now allow an analysis of both the method and experimentation presented within (Orriols-Puig and Casillas, 2010).

2.9.4 Problems identified in CSar

Following the discussion of the CSar algorithm, some potential problems have been found in the implementation and the methods of analysing the algorithm to alternative methods.

The size of intervals was address in the previous section on association rules, the CSAR algorithm uses a algorithm wide $maxInterval$ parameter which is set by the user. This does not seem to take into the account of the range of an attribute, therefore indicating that if this variable is not set correctly, the reviewed problem found in Brin et al. (1997b), where if the support for the consequent is high, the confidence will also be high. Leading from this it is not seen as infeasible that smaller intervals, in some rules could provide a more interesting rule and the combination of two intervals may reduce this interestingness. A simple example of both these problems, could be presented in the rule:

$$if\ age \in [20, 40]\ then\ went\ to\ glastonbury\ festival$$

.

to an expert this may not be that interesting, but the rule:

$$if\ age \in [35, 40]\ then\ went\ to\ glastonbury\ festival$$

could be more interesting, as the previous rule could have been overlooked as obvious. This could be dealt with by a smaller $maxInterval$ value, but this seems to take away the benefit of unsupervised learning if a parameter has to be tweaked so closely. Along with this if there is an attribute for an individuals salary, one value of $maxInterval$ would not be sufficient. It is thought that attributes in the CSar algorithm were preprocessed, to be within a range of 0 and 1, allowing a single interval variable to be set. However this does not seem possible whilst learning in an online style, as you will not see the data before it streams into the algorithm. Therefore this means the the range of values

will not be present, removing the ability to normalise attributes. Therefore XCSAR will have the ability to automatically set this parameter for each attribute, removing the need for the user to set it and addressing the problems identified.

Throughout the experimentation section within Orriols-Puig and Casillas (2010), a comparative study was only carried out against Apriori on datasets with categorical attributes. The datasets with numeric attributes were only deemed successful due to the system evolving rules that had high support and confidence, this is not seen as optimal. As rules with higher support and confidence could have been missed. Therefore as discussed in the further work section of Orriols-Puig and Casillas (2010), there needs to be a benchmark in terms of applying another algorithm to the datasets with numeric attributes.

Additionally, XCS in general is formulated as using a single step rule approach, in association rule mining this means there is only one item in the consequent. This provides a limitation to the rules that CSar can evolve, although it is seen as an optimisation in terms of runtime it can be argued that rules with more items in the consequent can be more interesting, suppose the following rule:

$$if\ age \in [20, 40]\ then\ buys\ cheese,\ smokes\ and\ walks \in [1, 5]\ miles\ a\ week$$

This rule highlights the extremely predictive nature of the age attribute, although this rule is obviously over emphasised.

Due to the overlapping problem being identified at a later date to the development of CSAR, it was thought that the CSAR algorithm would have similar problems. This is verified in Chapter4, further proving that the evaluation undertaken in Orriols-Puig and Casillas (2010) is not sufficient and therefore will be improved upon in Chapter 5.

2.10 Questions from background

This chapter has provided the foundations of the current state of both Association rules and Learning Classifier Systems, additionally it has provided direction for this research both in terms for theortical understanding and practical implementation advances. These questions therefore sumarise this learning and provide focus for this project:

1. Determination of the extent of the overlapping problem within XCS and CSAR
2. Gviing the ability to set the maxInterval parameter programmatically, allowing a more flexible and user friendly algorithm.
3. Further experimentation and evaluation that was not provided within Orriols-Puig and Casillas (2010)

2.11 Chapter Conclusion

Throughout this chapter there has been an introduction to both Association rule mining and Learning Classifier Systems. There has been an investigation into the alternative methods of mining association rules, along with this there has been discussion and analysis of what has been found to be the relative

advantages and disadvantages of mining using these methods identified in literature and by this author.

Additionally the problems recently found in the XCS algorithm and critique of CSar have been presented allowing analysis of how these can be improved upon and rectified in this project.

In conclusion, this background of association rule mining and Learning Classifier System has provided an excellent foundation to give the ability to design a robust algorithm and determine whether the output of such algorithm is useful and interesting to the user. Futhermore the problems identified within CSar and XCS in general give a great overview of the current situation of association rule mining within the field of XCS.

Chapter 3

Overview of Algorithm

This chapter provides and overview of the algorithm implemented for this project XCSAR (eXtended Classifer System for Association Rules), documenting the approaches taken in terms of quantifying fitness, the parameters that are stored and the operators for exploring the search space. Many of the approaches taken in this algorithm are inspired by CSAR with additional alternative mechanisms. It was deemed necessary to implement an algorithm similar to CSAR, to enable analysis against traditional quantitative rule miners which was not undertaken in (Orriols-Puig and Casillas, 2010) and source code did not accompany the paper. Additionally, CSAR is thought to have problems identified in Ioannides et al. (2011b), therefore improvements were needed to be carried out, along with this lack of analysis there were several mechanism not documented in the original paper.

The algorithm presented has the ability to mine 3 types of attributes; binary, categorical and quantitative. However binary attributes can be considered a categorical attribute as many uses for association rule mining, are only interested in whether items are bought with each other, rather than items not bought. In contrast to CSAR, XCSAR can be used to mine single and multi step consequents, where there is more than one item in the consequent. The implementation used a previous XCS implementation named XCSF a function approximator as a template, however due to the significant differences between mining association rules and function approximation many of the components had to be rewritten. Additionally the implementation of XCSAR was designed to allow the easy addition of alternative attribute types such as Temporal, therefore XCSAR was written in Java to use its object oriented nature. Furthermore, XCSAR is highly configurable presenting a configuration file which can control the fitness measures, population size, number of generation among others to decide the probability of the genetic operators.

3.1 Representation

Due the increase of variation of attributes that can be presented to the task of association rule mining, the typical tenary string representation of rules is not valid in this context. However due to the wealth of applications of XCS, there have been attempts of using continuous attributes within XCS notably XCSR presented in Wilson (2000). This algorithm takes a similar approach to XCSR in the handling of continuous attributes, in that they are represented by an interval value. An attribute represented in a rule, is given an upper and lower interval; $[l_i, u_i]$.

Additionally XCSAR makes use of micro and macroclassifiers, where if a rules is found using the GA and is already in the population the rules numerosity is increased. This numerosity is the number of microclassifiers of the parent macroclassifier or rule.

3.2 Algorithm Parameters

Many parameters are saved against each rule in the population, most of which are used to calculate fitness values. These include; *a*) Support *b*) Confidence *c*) Numerosity *d*) Whole rule matches *e*) Consequent matches *f*) Antecedent matches *g*) Timestamp *h*) Association/Action Set size (as) and *i*) Experience

Additionally for use in the GA, information about the distribution of both quantitative and categorical information is saved, this allows attribute values to be set when they are added to a rule in the GA operating schemes. For quantitative attributes the probability distribution along with other statistical summary information is stored such as the mean, minimum, maximum and standard deviation for the maxInterval parameter. This is collected over a user set range of generations, which can be varied in the hope to improving or changing the algorithms ability to learn online.

3.3 Genetic Search - Rule Discovery

The essence of any Genetic Algorithm is its genetic operators in terms of mutation and crossover, to allow exploration of the search space. Many different schemes are available to modify association rules, especially due to the varied attribute types that can be given to this algorithm. This section will document the various implemented schemes, that will be used within XCSAR.

Firstly it is important to discuss when the GA is triggered along with how and when rules are deleted from the population, when it is full. Triggering the GA is carried out the same as both XCS and CSAR, however this is known to cause problems with overlapping rules which will be further discussed in Chapter 4. The GA is triggered when the average number of generations since last GA was triggered, within this chosen action set, is greater than θ_{GA}.

When the population of rules surpasses the population size limit, the deletion mechanism is triggered. As in CSAR rules are deleted based on their association set size. CSAR also introduced measures, to increase the probability of less fit and experienced rules to be deleted using the following calculation:

$$dp_k = \frac{d_k}{\sum_{\forall_j \in \{P\}} d_j}$$

$$d_k = \begin{cases} \frac{as \cdot num \cdot F_{\{P\}}}{F_k} & \text{if } exp_k > \theta_{del} \text{ and } F_k < \delta F_{\{P\}} \\ as \cdot num & \text{otherwise} \end{cases}$$

Where dp_k is the rules deletion probability, $F_{\{P\}}$ is the average fitness of the population, θ_{del} and δ are user set parameters.

However it was found that this was not enough to delete all unwanted rules, as rules can be randomly generated through the GA, which can produce unfit rules increasing their numerosity, if they are already in the population. Due to these rules being unfit, they never get experience therefore never get given the higher deletion probability they deserve. Leading to the calculation to be modified to:

$$d_k = \begin{cases} \frac{as \cdot num \cdot F_{\{P\}}}{F_k} & \text{if } (exp_k > \theta_{del} \text{ or } (iteration - timestamp_k) > \theta_{old}) \text{ and } F_k < \delta F_{\{P\}} \\ as \cdot num & \text{otherwise} \end{cases}$$

This calculation therefore puts more pressure on low fit rules that are old or experienced, it is also hopeful that this will aid the online nature of this algorithm by deleting rules that have become unfit, due to dataset drift.

3.3.1 Mutation Schemes

Mutation schemes can be applied to both the rule and the values of each of the attributes in a rule, as will be presented.

Mutating Rule

Switching attributes Attributes in the antecedent and consequent can be switched, it is thought that this will not affect the parameters of the rule a great amount. As the itemset of the rule will not change, therefore this operator is seen to only modify the differences in how the rule is read, rather than changing the information content of the rule greatly.

Adding attributes Attributes are added to antecedent and/or consequent of the rule, attributes are only added to the consequent of the rule if the algorithm is in multi-step mode. The values are set to the attributes in the following ways dependent on the type of the rule. *a*) Binary attributes have there value set as true, as this is the only value that is valid in this type of attribute, *b*) Categorical attributes value is set based on a randomly selected value, where the probability of a value being selected is based on the probability distribution stored over the specified number of generations. *c*) Quantitative attributes have a value set based on the probability distribution of the attribute, then the interval of the attribute is generated based on randomly choosing a value to add and subtract from the random value chosen. This value is chosen from a range between 0 and $maxInterval/2$.

Mutating Values

Modifying Intervals If an quantitative attribute is chosen to be mutated then; the interval is randomly modified by adding or subtracting from both the upper and lower limit. The rules is also checked to ensure that the interval does not exceed the maximum interval. If the interval is deemed invalid, it is then fixed.

Modifying Categories If a categorical attribute is chosen to be mutated then a random value is chosen based on the probability distribution of that attribute, this is the same process as adding an attribute above.

3.3.2 Crossover Scheme

For crossover two parents are chosen from the action/association set, through roulette wheel selection based on the rules fitness, two offspring are cloned one from each parent. Each offspring has their

parameters reset and the time of creation set to the current generation.

The two offspring are then crossed over in turn, the rules antecedent and consequent are dealt with separately. Attributes from both rules are available for selection in the antecedent and consequent separately, with a random number of attributes selected for each side. The values for each of the attributes is kept the same as the parents.

3.4 Set Creation

3.4.1 Matchset Creation

Most LCS algorithms including CSar create a matchset based on whether or not the condition of the rule or in the case of association rules, the antecedent of the rule matches the given input. This technique is also applied in XCSAR.

3.4.2 Action/Association Set Creation

LCS inspired algorithms group the match set in terms of there action/consequent, one of the approaches documented in Orriols-Puig and Casillas (2010) used a similar technique. Additionally another technique was used by grouping based on the antecedent, these grouping schemes are documented in the background chapter 2.9.1. However it was found in Orriols-Puig and Casillas (2010), that creating the action set based on the consequent gave the best results therefore the same action set creation was used in XCSAR.

The selection of which action set was chosen for reproduction, using the following probability measure:

$$actionSelectionProb = \frac{totalActionSetFitness}{matchSetFitness}$$

3.5 Fitness Function

As discussed in the background chapter, there have been many options developed to quantify the interestingness of association rules. To allow comparison and the potentially provide solutions to some of the problems associated with this research, multiple fitness function were implemented which are outlined below. However due to the online nature of the implemented algorithm, there are difficulties when attempting to replicate the measures introduced. This is mainly due to the removal of traversing the dataset and producing accurate values for support, as this is not possible in online learning values for support have to be estimated from the given stream of data, hence some equations are modified slightly.

Due to the online nature of XCSAR, the calculations of support and therefore confidence are slightly different to traditional calculations. For each rule counts are stored determining the number of times the consequent, antecedent and whole rule were matched. Respectively these parameters are cm, am and wm, this notation allows for the calculations for support and confidence to be presented.

$$supportWholeRule = \frac{wm}{currentGen - timestamp}$$

$$supportConsequent = \frac{cm}{currentGen - timestamp}$$

$$conf = \frac{supportWholeRule}{supportConsequent}$$

It should be noted that these values are estimates of true support and confidence as not all instances are seen, additionally rules are not always found from the first generation, so a rules support and confidence will generally not be calculated from the first instance given to the algorithm.

XCS This fitness function is as close as possible to the standard XCS fitness function including the fitness sharing aspects. However due to the applications of XCS, the typical prediction error calculation is not available. Therefore the calculation for XCS fitness is:

$$XCSfitness \leftarrow XCSfitness + \beta((conf * numerosity)/actionSetFitness) - XCSfitness)$$

Confidence The standard confidence measure was used as a fitness measure, however it was found that this favoured rules that in some cases were not significant due to their low support.

Support and Confidence This fitness function as used in CSAR is calculated as:

$$fitness = (supportWholeRule * conf)^d$$

Where d allows the user to push the algorithm to the direction of highly fit rules.

The use of a pareto optimal fitness function where multiple objectives are used, was also explored. However this was found to be infeasible for a multitude of reasons of which included, the computation of the rank of a rule to determine the pareto front was infeasible due to the large number of rules. Additionally the widespread use of support and confidence and the ease of comparison against other algorithms, was found to be the best fitness function.

3.6 Covering Operators

An advantage of using an LCS inspired algorithm, is the lack of need to populate the population with rules at the start of the algorithm. This not only reduces the start up time, but also ensures all rules in the population meet at least one instance of the dataset. The covering operator is executed, when there are not enough instances in the match set, the minimum number of rules in the match set is set by the user. In this algorithm two methods were used for covering outlined below.

Every Combination When mining for association rules using Apriori frequent itemsets are mined, at this point all possible combinations of the itemset are checked for having high confidence. The same thought process was considered in XCSAR, however only one attribute was given to the consequent, in an attempt to not add too many rules into the population. Therefore this covering operator took every attribute from the input which had a value, creating N rules where N is the number of attributes with values. The antecedent was filled with X attributes, chosen randomly from 1 to

$N-1$. The unused attributes were then picked until, X attributes had been added to the antecedent. Each rule was checked for an identical rule in the population, if this happened the numerosity of that rule was increased by 1.

Random The second option for covering was a more random approach, additionally giving the ability to put more than one attribute in the consequent. Then number of rules to be added was chosen randomly from Y, the number of rules that need to be added to the matchset calculated as minimum number of matched rules minus the number of matched rules to N as defined above. The same process for choosing attributes for the antecedent described above was then used for both the antecedent and consequent, ensuring that there was always at least one attribute in both sides.

3.7 Max Interval Setting

A critique of the CSar algorithm was the need for the user to set a maxIterval parameter, it was not explicitly expressed in the paper but CSar used or preprocessed quantitative attributes to have values between 0-1. This is considered to be infeasible whilst learning online, due to the inability to normalise attribute values that have not yet been seen. Therefore the maxInterval parameter was set during runtime, automatically for each attribute rather than one system wide parameter as used by CSAR. There was some experimentation with measures of spread for each attributes, however it was found that setting the maxInterval parameter to 2 times the standard deviation, produced satisfactory results in that the intervals were not too wide, whilst giving the ability to capture the spread of an attribute using a statistically sound measure.

Additionally the argument of using statistical techniques to describe rules has been researched and reviewed in the background chapter, where Aumann and Lindell (1999) argued that association rules should use statistics to determine intervals. This approach has in part used this wisdom, whilst applying it to be used in an online nature and within a GA removing the restrictions imposed with Aumann and Lindell (1999).

3.8 Rule Reduction

Reducing the number of rules presented to the user is highly important in association rule mining, due to the general excessive nature of rule output. Additionally whilst mining quantitative attributes, there is also a chance of creating very similar rules with only slightly different intervals. Therefore a mechanism was developed to reduce the number of rules when mining quantitative attributes.

Firstly rules were grouped together if all the attributes were the same in both antecedent and consequent, this was initially carried in a $O(n^2)$ fashion by choosing a rule from the population and finding all other rules with the same attributes. However this was found to be inefficent, so rules were sorted based only their attributes without values. Therefore grouping could easily be done in $O(nlogn)$ time complexity, where n is the number of rules in the population.

Once the grouping was carried out, if any intervals overlapped the rule with the highest fitness was chosen and the rest were deleted from the population. This allowed a significantly reduced ruleset to be produced to the user and analysed in Chapter 5. Additionally to have a greater confidence in the support and confidence values, any rule with experience less than 5000 was removed. Experience

is the number of times the rule has been in the match set, therefore the greater amount of times this takes place the more times fitness will be updated, therefore having a higher probability of have the correct counts stored.

3.9 Chapter Conclusion

This chapter has presented the XCSAR algorithm developed for this project, this will now allow an indepth analysis of the overlapping problem documented in the background chapter, before further evaluating the XCSAR algorithm in the chapter there after.

Chapter 4

Analysis and Improvements

Throughout this chapter the problems that have been identified within the XCS and CSAR algorithms, will be analysed and improvements presented and evaluated. This chapter will contribute to the theoretical, implementation and investigation content of this research, allowing further analysis of the algorithm using differing datasets and comparison against traditional methods in the next chapter.

4.1 Overlapping Rules

4.1.1 Description of Problem

Section 2.8 reviews the background of the problem of overlapping rules, including reviewing the initial evidence and the extent of the problem presented in (Ioannides et al., 2011b). This section will describe the problem with an example, before providing evidence of the problem using traditional XCS configuration, then moving to the configurations used in the CSAR algorithm.

To allow a general understanding of this problem, it is necessary to define it succinctly. In this project the problem is deemed to be present when rules within the population have lower numerosity than expected or compared to other rules that overlap similar instances from the dataset, but have similar fitness. This low numerosity in extreme cases, can cause that rule to be deleted from the population and therefore not provided to the user.

It has been identified that there are at least three contributions in the XCS algorithm causing this problem, these are discussed using Figure 4.1 as an example of overlapping rules in a function approximation learning task, as this is easiest to visualise.

Relative Accuracy Fitness Function XCS updates fitness towards relative accuracy, this involves sharing fitness within a niche or action set. Which leads to the problem of rules taking over these niches and competing other rules out of the population, due to their exceptionally high fitness measure. Relative fitness also includes numerosity in the calculation, therefore once a rule becomes dominant its fitness can exponentially grow.

Niche GA The niche GA within XCS is thought to provide substantial constraints for overlapping rules. Figure 4.1, shows rules overlapping in a 2D search space it can be seen that the red rule overlaps with all other rules. This means that at inputs X=[1,2] and Y[1,3] or Y[3,4.5], the red rule will have to compete with each of the other rules for reproduction. However all other rules, will not have competition to reproduce at inputs other than the overlapping regions. This would potentially allow for their numerosity to increase and increase their chances of surviving in the population.

GA Trigger The GA trigger within XCS and CSAR is a user set parameter to determine when the GA should be ran. When the average time since the last GA has been triggered in the actionset is greater than θ_{GA}, the GA is triggered. This affects the overlapping issue when an input is in the overlapping area, it is likely that a rule other than the red rule has been in the GA. Therefore this would reduce the average time since the last GA and reducing the chances of the GA being triggered. This therefore has a similar problem to both the problems above, in that the lack of GA triggering causes less copies of a rule to be created. To further expand this if an input is given at (3,2) and the green rule has not been in the GA for a long time, then the GA will be triggered. If the next input is (1.5, 2), as the green rule had only just triggered the GA it would bring down average time of that actionset with both the red and green rule present. Therefore the GA may not be triggered, causing the red rule to loose its chance of reproduction.

Table 4.1a shows the dataset used in the traditional rule mining example in Appendix A, additionally Table 4.1 shows some rules from this dataset that overlap and therefore would be in the same match set and have the potential to compete with each other. To allow simplicity only rules with one consequent are used, additionally CSAR was only developed to mine rules with one consequent.

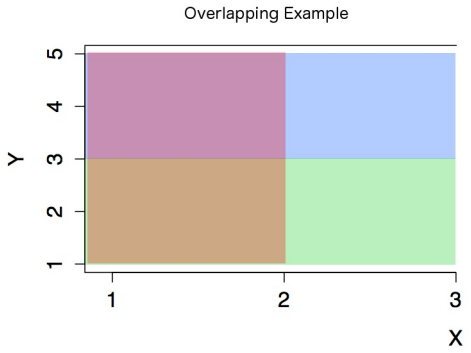


Figure 4.1: Example of overlapping regions in a 2D space, with 3 rules red, green and blue

Cereal	Bacon	Milk	Bread	Butter	Doughnuts
1	1	1	0	1	0
1	1	0	1	1	0
1	0	1	0	0	1
1	0	1	1	1	0
1	0	1	1	1	0

(a) Transaction Database

Rule	Conf
$\{Milk\} \rightarrow \{Cereal\}$	1
$\{Butter\} \rightarrow \{Cereal\}$	1
$\{Milk\} \rightarrow \{Butter\}$	0.75
$\{Cereal, Milk\} \rightarrow \{Butter\}$	0.75

(b) Overlapping rules, when an input transaction has Milk and Butter present

Table 4.1: Transactions and Overlapping Rules

One particularly important point to note is that the first two rules presented in Table 4.1, will all be in the same action set and therefore will compete to be chosen for reproduction. These rules are non-subsuming, meaning they should always be presented to the user. However rules three and four are subsuming, therefore there can be an argument that the most general rule of $\{Milk\} \rightarrow \{Butter\}$, is the most important to be presented to the user. This should be taken into consideration, although exhaustive methods such as Apriori, will find all these rules and it can be argued that having more

specific rules that are fit are useful to the user, leading to the need to investigate both subsuming and none subsuming competition.

4.1.2 Evidence of Problem

Initially it is important to show evidence of this problem, using traditional XCS learning styles such as the fitness function, niche GA and GA trigger. It is thought that these factors are the biggest contribution to this problem, therefore will have the most significant impact on interesting rules not presented to the user or having substantially lower numerosity than expected. Alternative methods presented in CSAR will then be explored to determine whether the removal of fitness sharing, will rectify the problem or if there is a need for more extensive changes.

To allow for easy evaluation of the overlapping issue, a small dataset will be used as presented above. This enables an easy understanding of what rules are processed into what action/association set, it also allows the determination of what rules are missing more easily. Additionally, there will be tests on a range of population sizes this allows a simulation of a larger problem size, which would not fit into any population due to computational costs. It maybe noted that setting a population small is unrealistic, however when the number of attributes grows within a dataset, it soon becomes computationally infeasible to provide a large enough population. For example a dataset with 20 attributes has 1,048,576 itemsets and 3,484,687,250 possible association rules, when allowing multiple items in the consequent. It is therefore necessary to attempt to simulate the problem of a population becoming full and numersities falling to 0, hence causing a rule to be deleted.

This reduction in population size has not been studied in any other publication, therefore it is also a contribution to this research and an attempt to fully understand an LCS algorithm at capacity, without having to analyse populations of rules within the tens of thousand, which is infeasible to effectively compare and evaluate.

XCS Configuration

The XCS configuration used in this section, is the standard XCS match set and action set creation and the fitness function shown in Section 3.5. In addition to the standard XCS configuration, the deletion vote calculation presented in Chapter 3 will be used, this is due to the substantial problems found with the traditional XCS and CSAR calculation.

Firstly as discussed the dataset presented in Appendix A will be used, with a population limit of 200 and 50000 generations. It should be noted that this population size, is beyond the amount of rules within this dataset, which would be 186 with one attribute in the consequent. To determine which rules should be in the population, Apriori was run with $minSupport$ set to 0.5 and $minConfidence$ was set to 0.5. Finally this produced 22 rules, further proving that there should be more than enough room to accommodate the fittest rules within this dataset.

The algorithm was then executed on the dataset, storing numerosity values for each rule every generation. At the end of the 50000 generations, the output from Apriori was read and compared against these numerosities values. Throughout every test all rules from Apriori were found in at least in one generation, hence proving the algorithm had the ability to find the maximally fit rules. Additionally, it was found that all confidence measures were the same as found using Apriori, this

result was the same as found in Ioannides et al. (2011b), where the reward and error values of a rule were correct.

However although all rules were found in at least one generation, not all rules were still in the populations at the end of 50000 generations. In fact 6 of the 22 rules found by Apriori, were not present in the final population. These rules are shown in Table 4.2.

Rule	Support	Confidence
$\{Butter = 1\} \rightarrow \{Cereal = 1\}$	80	100
$\{Butter = 1, Milk = 1\} \rightarrow \{Cereal = 1\}$	60	100
$\{Cereal = 1\} \rightarrow \{Butter = 1\}$	80	80
$\{Butter = 1, Cereal = 1\} \rightarrow \{Bread = 1\}$	60	75
$\{Butter = 1, Cereal = 1\} \rightarrow \{Milk = 1\}$	60	75
$\{Cereal = 1, Milk = 1\} \rightarrow \{Butter = 1\}$	60	75

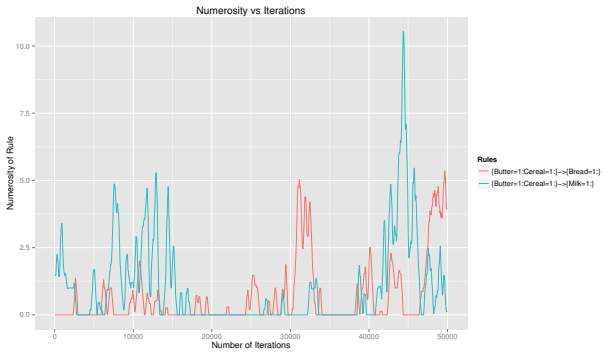
Table 4.2: Rules not present using XCS Configuration @ PopSize=200, Generations =50000

Figures 4.2a, 4.2b and 4.2c show that many of these rules gain significant numerosity, but are then deleted from the population during the course of the algorithm. This behaviour is obviously not wanted as it would be more intuitive for rules to be discovered and kept in the population, if they are deemed fit enough. This is essential as from the figures presented, it would be by chance that choosing a certain number of generations would cause the rule to be present in the final population. Further investigations into why these rules were not in the final population, was carried out by comparing the fitness of the rules to others potentially in the same action set. This was carried out on the rules with Cereal and Butter in the consequent, as there were two rules for each not in the final population.

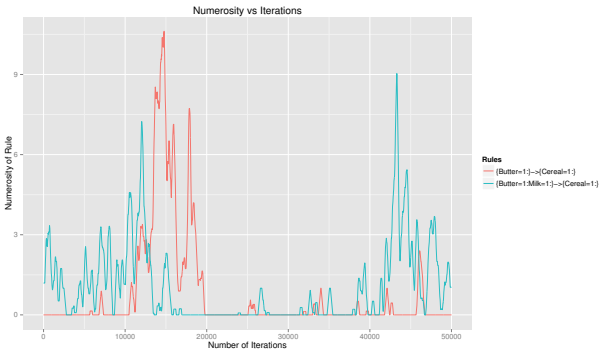
Figure 4.3a, shows the rolling mean of fitness over a 150 generations for all rules with Cereal in the consequent. These rules would generally be in the same action set, dependent on whether the antecedent matched the input. It can be seen very clearly here that, the rule $\{Milk\} \rightarrow \{Cereal\}$ dominates this niche. This is intuitive as this rule, has high support and confidence at 80% and 100% respectively. However, the rule $\{Butter\} \rightarrow \{Cereal\}$ has the same values, but has significantly lower fitness and therefore numerosity. A reason for this could be the increase in competition for rules with Butter in them, from Appendix A, it can be seen that Butter is in more frequent itemsets. This means there is a lower probability that this rule would be chosen for reproduction, due to more rules being in the action set. These regions have many overlaps causing extreme competition, where some rules loose out and have their fitness significantly reduced due to fitness sharing. Emphasis should be made that these rules are non-subsuming, hence further showing that due to the high fitness of both they should be presented to the user.

The same results can be seen in 4.3b, where the rule $\{Bread\} \rightarrow \{Butter\}$ can be seen to dominate the niche for most generations. The justification for this is similar to Butter being in more frequent itemsets, whereas Bread is in less frequent itemsets therefore when bread is given as an input, there are less rules to compete for reproduction causing the rule $\{Bread\} \rightarrow \{Butter\}$ to be chosen more often.

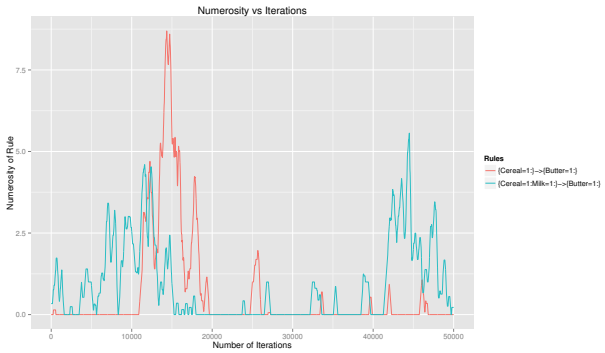
Similar results to this have been explored in a recent paper Kovacs and Tindale (2013), where rules that are more general gain a generality bonus, due to these rules being present in more action sets. Due to the lack of tenary encoding within this algorithm, it can be hard to see when a rule is general.



(a) Bread and Milk in the Consequent



(b) Cereal in the Consequent



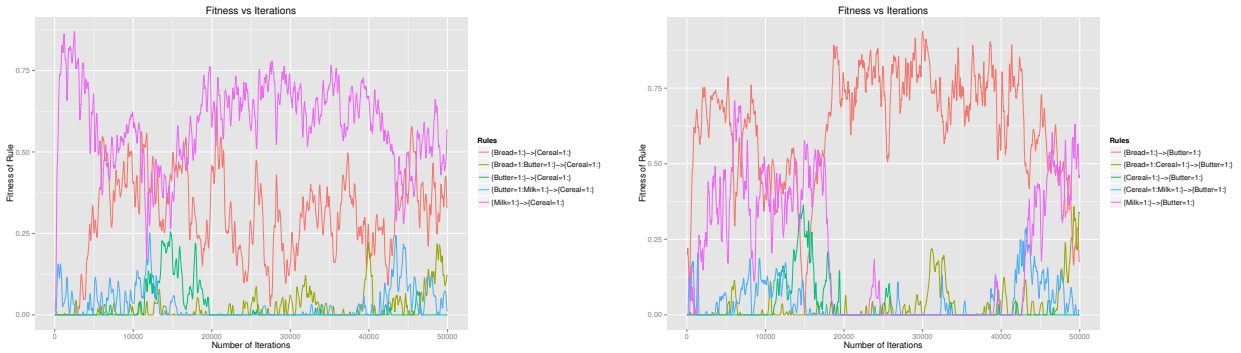
(c) Numerosity of rules with Butter in the Consequent

Figure 4.2: Numerosity vs Iterations for rules that are not provided in the final population. It can be seen that, all of the rules are discovered and some rules have considerable numerosity until it falls to 0 and are deleted from the population

However a binary association rule can be converted to a standard XCS tenary representation, as shown in Table 4.3. Hence proving that the rule shown is in fact a general rule, therefore having the advantage of a generality bonus. As a side note, a rule is more general in tenary encoding when more # values are present, as this means that attribute can match any input. However within association rules due to the large amount of rules provided to the user, this bonus is not as simple as the typical XCS algorithm. General rules within association rule mining will have a generality bonus, although this could be offset by the amount of rules that are within the action set causing more competition. Table 4.4, shows that general itemsets can have the same support as larger more restrictive itemsets, therefore reducing the generality bonus of the most general itemsets and causing more rules to be present in an action set. An example is $\{Milk\}$ and $\{Milk, Cereal\}$, which although the prior is more general in terms of traditional tenary encoding both itemsets have the same support.

Rule	Tenary Representation { C,B,M,Br,D → Bu }
$\{Bread\} \rightarrow \{Butter\}$	###1# → 1
$\{Cereal, Milk, Bread,\} \rightarrow \{Butter\}$	1#11# → 1

Table 4.3: Conversion from association rules representation, to traditional tenary strings used in XCS. This shows that rules with more items are less general, therefore are less likely to be in the match set. This causes a generality bonus for rules with less items, although as discussed this can be reduced with less general rules having similar support as more general rules.



(a) Cereal in Consequent

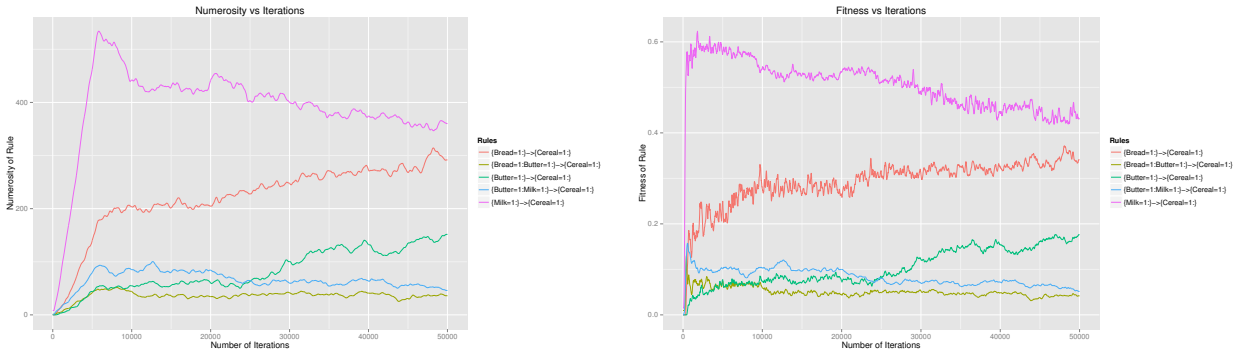
(b) Butter in Consequent

Figure 4.3: Fitness vs Iterations for all rules that were found in Apriori and XCSAR with Cereal or Butter in the consequent. This shows the domination in terms of fitness, especially within the rules with Butter in the consequent. Hence providing justification for the lower numerosity, of the rules not found in the final population.

tid	*Support*
$\{Cereal, Milk\}$	0.8
$\{Cereal, Bread\}$	0.6
$\{Cereal, Butter\}$	0.8
$\{Milk, Butter\}$	0.6
$\{Bread, Butter\}$	0.6
$\{Milk\}$	0.8
$\{Cereal\}$	1
$\{Milk\}$	0.8
$\{Bread\}$	0.6
$\{Butter\}$	0.8

Table 4.4: Itemsets and their support, it can be seen that some itemsets of size 2 have similar support measures as itemsets of size 1. Therefore lowering the generality bonus of rules that are more general (have smaller itemsets in the antecedent)

To determine the affect of population size on the overlapping problem, the population size was expanded from 200 to 500 and 5000. A population size of 500, still caused rules not to be included in the final population as shown in Table 4.5. However when increasing the population to 5000 all rules found by Apriori, were also found using this configuration. The numerosities and fitness of the rules with Cereal in the consequent can be seen in Figures 4.4a and 4.4b, which show similar patterns as found with lower populations sizes. They can be seen as much steadier, although some rules still dominate the niche causing rules with similar confidence levels having significantly lower fitness and numerosity values. This further proves the intuition that fitness sharing unfairly impedes overlapping rules, additionally it shows that when a population becomes full it is more likely that the algorithm will become unstable with rules being found and deleted, when they should be present in the population consistently, so as to reduce the random affects upon whether a fit rule would be present in the final population.



(a) Numerosity vs Iterations

(b) Fitness vs Iterations

Figure 4.4: Fitness and Numerosity values for rules with Cereal in the consequent, meaning these rules should often be in the same action set and therefore compete against eachother for the chance to reproduce. These results are produced with population size of 5000, which greatly improves the stability of the algorithm as numerosities of rules are not reduced to the point of deletion.

Rule	Support	Confidence
$\{Butter = 1\} \rightarrow \{Cereal = 1\}$	80	100
$\{Bread = 1, Butter = 1\} \rightarrow \{Cereal = 1\}$	60	100
$\{Bread = 1, Cereal = 1\} \rightarrow \{Butter = 1\}$	60	100
$\{Cereal = 1\} \rightarrow \{Butter = 1\}$	80	80

Table 4.5: Rules not present using XCS Configuration @ PopSize=500, Generations =50000

In conclusion it has been found that the XCS configuration has substantial problems when rules overlap. The generality bonus found in Kovacs and Tindale (2013), still has an affect in association rule mining. However this can be offset if there are other rules that are less general but have high support, as this increases the size of an action set and therefore competition. It was also found that although pruning does not take place in the same way as Apriori, there is an implicit pruning in that rules with less support will be in less match sets and therefore less likely to be selected for reproduction. Fitness sharing was shown to be the main cause of some rules becoming highly dominant within a niche, severely impeding other rules both in terms of fitness and numerosity. Throughout the next section the CSAR configuration will be explored, which has the benefit of not using fitness sharing.

CSar Configuration

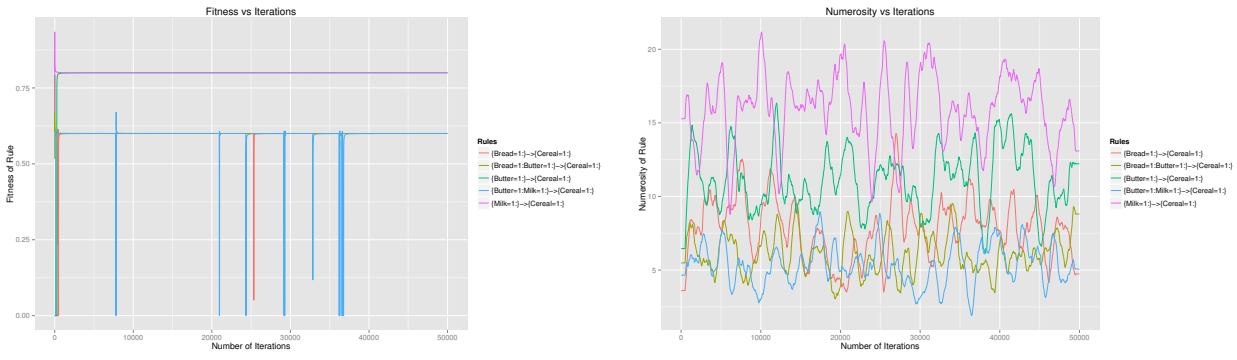
To further investigate the affects of overlapping rules, it was necessary to use the CSAR configuration and determine if the results presented within Orriols-Puig and Casillas (2010) could be improved upon, in light of the overlapping issues found within XCS. There are some substantial differences between a vanilla XCS implementation and the one presented in (Orriols-Puig and Casillas, 2010). The main difference is the fitness function calculation and update, the CSAR fitness function is presented in Chapter 2, which introduces support into the fitness measure. More importantly CSAR does not use relative fitness updates, therefore fitness is not shared within the action Set. Additionally fitness updates are propagated to the whole match set and not limited to the action set. This section will focus on determining whether CSAR has the same problems found above, along with analysing

the affects of removing fitness sharing. This analysis and further understanding will be an additional contribution from this project, aiding the understanding of overlaps and determining the main cause. As CSAR removes the first cause listed above, it enables the ability to further explore the remaining causes.

Whilst running XCSAR with the CSAR configuration, the same population sizes were used as above. Figures 4.5a and 4.5b, show the fitness and smoothed numerosity, which shows at certain generations of the algorithm the rules $\{Butter = 1, Milk = 1\} \rightarrow \{Cereal = 1\}$ and $\{Bread = 1\} \rightarrow \{Cereal = 1\}$ are deleted depicted by the drop to 0 fitness. This behaviour is undesirable as whether these rules are included in the final population is highly dependent on the number of generations the algorithm is ran until. This behaviour would be more acceptable if rules were deleted near the start of the algorithm, however fitness can be seen to plummet at many different points of the algorithm.

The fitness values can be seen as very stable, where it is almost always calculated as the support of the rule due to confidence of the maximally fit rules being 1. However this stability due to the removal of fitness sharing, does not seem to completely solve the problem of overlaps due the some rules being deleted from the populations and lower numerosities values of rules with similar fitness. As can be from Figures 4.6a and 4.6b, the rule $\{Cereal = 1\} \rightarrow \{Butter = 1\}$ dominates the niche, this rule does have the highest fitness by 0.046 or 7.8% on average. However this does not warrant the 71.83 or 56.9% increase on of numerosity compared to rules with slightly lower fitness, however the rule $\{Cereal = 1\} \rightarrow \{Butter = 1\}$ does have significantly higher support for its antecedent at 1 further aiding its generality bonus.

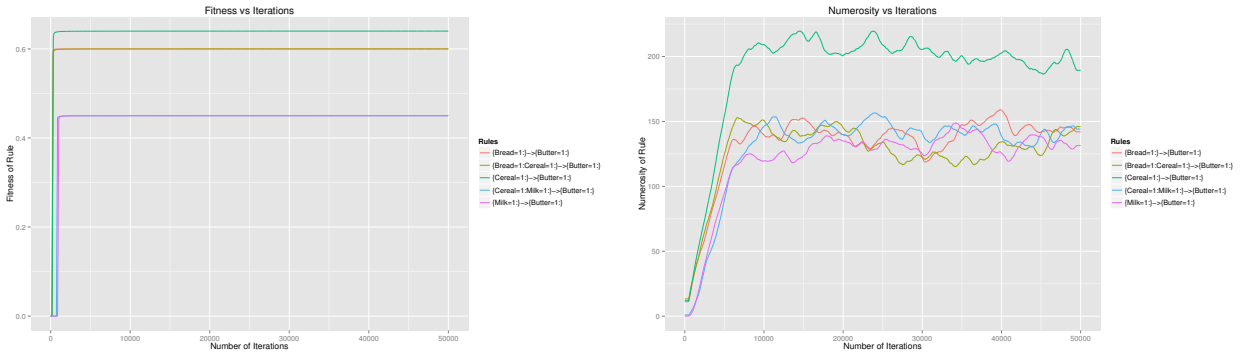
Additionally it was expected that rules, $\{Bread = 1\} \rightarrow \{Butter = 1\}$ and $\{Bread = 1, Cereal = 1\} \rightarrow \{Butter = 1\}$, would be easily distinguishable in terms of numerosity than the other rules at significantly lower fitness. These rules were also expected to have similar numerosity, but as can be seen from Table 4.6, $\{Bread = 1\} \rightarrow \{Butter = 1\}$ does not have significantly higher fitness but does have significantly higher numerosity at a significance level of 95%. This further suggests rules gain a generality bonus as previously stated, even if the antecedents of these rules have the same support at 0.6.



(a) Fitness vs Iterations

(b) Numerosity vs Iterations

Figure 4.5: Fitness and Numerosity values for rules with Cereal in the consequent, using the CSAR configuration and a population of 200. It can be seen that some rules are deleted during certain generations of the algorithm causing instability in the algorithm and unwanted deletion of those rules. Numerosity vs Iterations was smoothed over 1000 iterations, due to extensive changes in numerosity.



(a) Fitness vs Iterations

(b) Numerosity vs Iterations

Figure 4.6: Fitness and Numerosity values for rules with Butter in the consequent, using the CSAR configuration and a population of 5000. It can be seen that $\{Cereal = 1\} \rightarrow \{Butter = 1\}$, dominates the niche even though fitness is on average 0.046 greater than the rules at the next fitness level.

Rule	Avg. Fitness	p-value
$\{Bread = 1\} \rightarrow \{Butter = 1\}$	0.5907	0.46
$\{Bread = 1, Cereal = 1\} \rightarrow \{Butter = 1\}$	0.5905	0.46

(a) Comparing the average fitness over 10 runs of the algorithm, which gives no significant difference between fitness of the rules.

Rule	Avg. Fitness	p-value
$\{Bread = 1\} \rightarrow \{Butter = 1\}$	132.16	0.041
$\{Bread = 1, Cereal = 1\} \rightarrow \{Butter = 1\}$	126.14	0.041

(b) Comparing the average numerosity over 10 runs of the algorithm, which gives a significant difference at a level of 95%

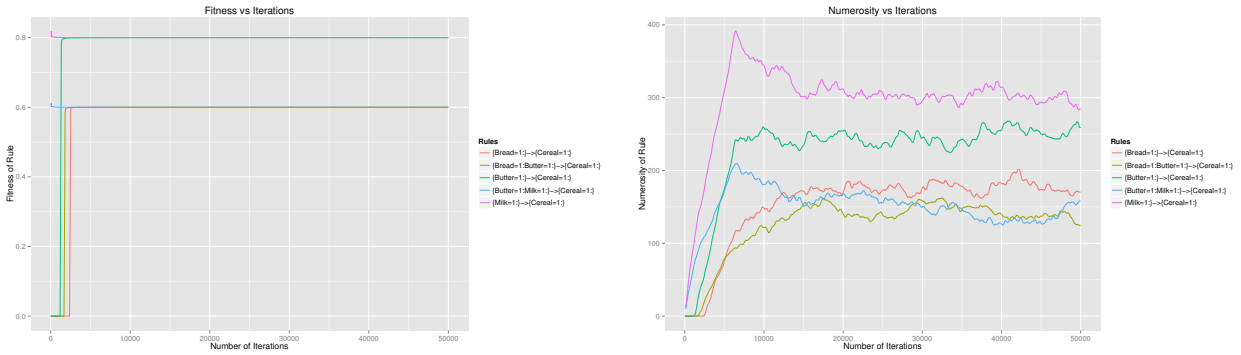
Table 4.6: Testing the difference between the fitness and numerosity values of the rules $\{Bread = 1\} \rightarrow \{Butter = 1\}$ and $\{Bread = 1, Cereal = 1\} \rightarrow \{Butter = 1\}$.

The previous results have shown the generality bonus and the overlapping problem with rules that subsume each other, however rules that do not subsumed are very important to analyse as these rules are always wanted by the user. Therefore, figures 4.7 show the fitness and numerosity values for rules with Cereal in the consequent, it can be seen that rules $\{Butter = 1\} \rightarrow \{Cereal = 1\}$ and $\{Milk = 1\} \rightarrow \{Cereal = 1\}$ have the same fitness and support for the antecedent, although the numerosity for $\{Milk = 1\} \rightarrow \{Cereal = 1\}$ is considerably higher as show in Table 4.7.

Rule	Avg. Numerosity	p-value
$\{Milk = 1\} \rightarrow \{Cereal = 1\}$	289.24	6.674e-08
$\{Butter = 1\} \rightarrow \{Cereal = 1\}$	239.94	6.674e-08

Table 4.7: Comparing the average numerosity over 10 runs of the algorithm for rules with Cereal in the consequent, which gives a significant difference at a level of 95%. Showing the rule $\{Milk = 1\} \rightarrow \{Cereal = 1\}$ has significantly higher numerosity, even though all fitness measurements are the same.

In conclusion there is substantial evidence that the problem identified in Ioannides et al. (2011b) is in fact a problem in CSAR in both subsumed and non-subsumed rules. It is also thought that any other algorithm following typical XCS matching schemes and using a niche GA, will also have



(a) Fitness vs Iterations

(b) Numerosity vs Iterations

Figure 4.7: Fitness and Numerosity values for rules with Cereal in the consequent, using the CSAR configuration and a population of 5000. It can be seen that the two rules with the highest fitness, have the same fitness however their numerosity is significantly different

this same problem. Additionally the problem can have even larger affects, due to the process of association rule mining and the desire to output many rules. Furthermore although not explicitly shown, rules with other types of attributes such as categorical and numeric further expand the search space justifying the need to explore solutions to ensure all maximally fit rules are kept in the population. These other types of attributes not only cause an increase in the search space, but also the size of the action and match sets, causing more competition within them.

4.1.3 Potential Solutions

In the previous section it was proven that aspects of both XCS and CSAR, contribute to the detrimental reduction of numerosity and sometimes deletion of fit rules, that are wanted in the final population. In this section potential solutions will be evaluated to determine whether this affect on overlapping rules, can be reduced or removed completely. There will be a focus on the identified causes presented at the start of this chapter, it was found that removing fitness sharing in the CSAR configuration had positive affect on the stability of the algorithm, so this will be used as a starting point for evaluation. An ideal solution would find all maximally fit rules and they would not be deleted from the population.

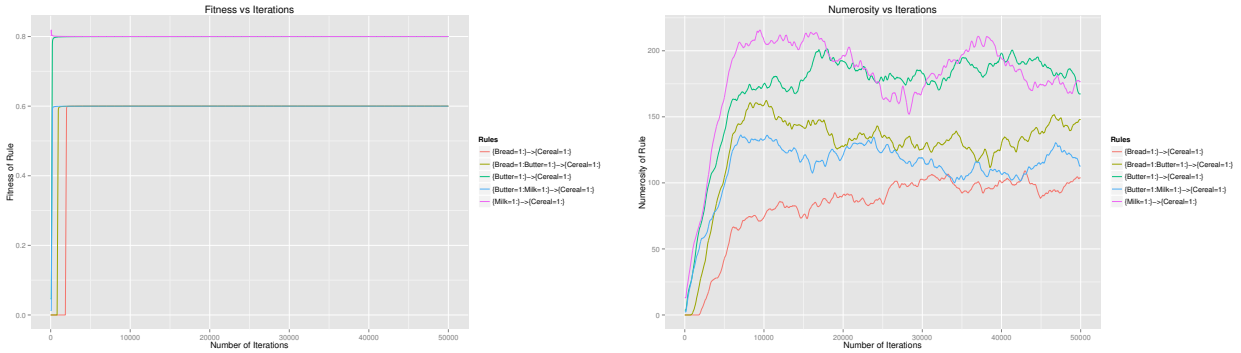
Additionally as discussed the size of the population has a large affect on rules being deleted and due to the search space growing exponentially, through the introduction of attributes and different types of attributes the size of the population often cannot be expanded to fit these rules. Therefore a solution would also have a high chance of producing all maximally fit rules with a very small population, once again simulating a larger search space filling a population to capacity. However it should also be noted that due to the heuristic and stochastic nature genetic algorithms, this cannot happen on every execution of the algorithm. Therefore multiple runs of the algorithm are needed to robustly evaluate these solutions.

Panmictic GA

It was found that much of the competition came due to the selection of rules using a niche GA, only selecting rules from the action set for reproduction, this introduced extreme competition within these

niches. It is therefore logical that reducing this competition within such a small subset of the rule population, has the potential to ease this competition. The use of a panmictic GA, where all rules in the population have the potential to be selected for reproduction can be used, instead of a niche GA. This solution was also applied in Kovacs and Tindale (2013), with success.

Figures 4.8 shows the fitness and numerosity of the same rules identified in the previous section that were impeded by overlapping. It can be seen that the use of a panmictic GA, has dramatically helped the problem. This is especially true for the non-subsumed rules $\{Butter = 1\} \rightarrow \{Cereal = 1\}$ and $\{Milk = 1\} \rightarrow \{Cereal = 1\}$ where their numerosity is very similar and inline with their fitness. While comparing numerosity over 10 runs using different random seeds, did reduce the significance of the difference, the problem was not fully rectified therefore there is a need to further investigate the last cause.



(a) Fitness vs Iterations

(b) Numerosity vs Iterations

Figure 4.8: Fitness and Numerosity values for rules with Cereal in the consequent, using the CSAR configuration and a population of 5000, using a panmictic GA.

Rule	Avg. Numerosity	p-value
$\{Milk = 1\} \rightarrow \{Cereal = 1\}$	161.93	0.0089
$\{Butter = 1\} \rightarrow \{Cereal = 1\}$	179.84	0.0089

Table 4.8: Comparing the average numerosity over 10 runs of the algorithm for rules with Cereal in the consequent, which gives a significant difference at a level of 95%

There is also a need to learn how the panmictic GA affects the generality bonus found in the last section, this can be analysed using the rules with Butter in the consequent. Table 4.9, shows that the generality bonus has decreased when using a panmictic GA the rules numerosity is now the opposite way round. It was found that this generality bonus was much larger than initally anticipated, due to the mutation scheme working in the opposite direction of the generality bonus. The mutation scheme was causing genetic drift towards less general rules, due to the probability of attributes being added and removed from the antecedent of the rule.

To explain this fully an example will be used, if the rule $\{Bread = 1\} \rightarrow \{Butter = 1\}$ is chosen for reproduction and this rule is also chosen to be mutated, firstly attributes will be added to the antecedent randomly. The algorithm chooses how many rules to be added from the attributes not currently in the rule, for example this rule has 4 attributes that are not currently present. Therefore the algorithm has a choice of adding between 0 and 4 attributes, therefore giving the rule a probability of 0.8 additional attributes will be added. Once the chosen amount of attributes have been added,

the rule goes through the process of removing attributes. The amount of rules to be removed from the antecedent is between 0 and the number of attributes minus one ensuring there is at least one attribute in the antecedent. If in the example one attribute is added to the rule and it is randomly chosen as Milk, giving the rule $\{Bread = 1, Milk = 1\} \rightarrow \{Butter = 1\}$ there would be a probability of 0.5 that an attribute would be deleted. This value is lower than the probability of adding an attribute, therefore causing the algorithm to drift towards less general rules. This shows the previous power of generality bonus and the affect the use of a panmictic, has as the less general rules now has significantly higher numerosity.

However this is not wanted for subsuming rules in that general rules are generally easier to read than longed rules more specific rule, additionally there has been much investigation in ensuring that when rules have similar fitness their numerosity is also similar. Therefore to fix this new problem found, which has the potential of being present in other XCS applications where attributes are added and deleted especially CSAR a new mechanism was introduced. Where a probability was given for both adding and deleting rules, which can be set by the user, giving the ability to control the length of the rules produced by the algorithm, although it is thought shorter non-subsuming rules will generally be preferred.

Rule	Avg. Numerosity	p-value
$\{Bread = 1\} \rightarrow \{Butter = 1\}$	108.54	4.76E-05
$\{Bread = 1, Cereal = 1\} \rightarrow \{Butter = 1\}$	128.6	4.76E-05

Table 4.9: Comparing the average numerosity over 10 runs of the algorithm, which shows a significant difference at a level of 95%

Removal of GA Trigger

The GA trigger has been identified as providing another cause of the overlapping problem, hindering overlapping rules ability to take part in reproduction therefore not allowing the ability to increase their numerosity. To test this concept the GA trigger was set to 0, in affect removing the trigger and causing the algorithm to always evolve rules whilst still using the panmictic GA. From Figure 4.10, it can be seen that the removal of the GA trigger also has a positive affect on the overlapping problem, showing that the top rules now do not have significantly different numerosity values, shown in Table 4.10. Additionally the other rules can be seen to be much more grouped, also inline with their fitness values.

Rule	Avg. Numerosity	p-value
$\{Milk = 1\} \rightarrow \{Cereal = 1\}$	286.78	0.11
$\{Butter = 1\} \rightarrow \{Cereal = 1\}$	279.33	0.11

Table 4.10: Comparing the average numerosity over 10 runs of the algorithm for rules with Cereal in the consequent, which does not givs a significant difference at a level of 95%

Larger Dataset

To ensure that these results were not biased by this simple dataset, the configurations were ran on the adult dataset including only the categorical attributes. To allow easy evaluation the number of

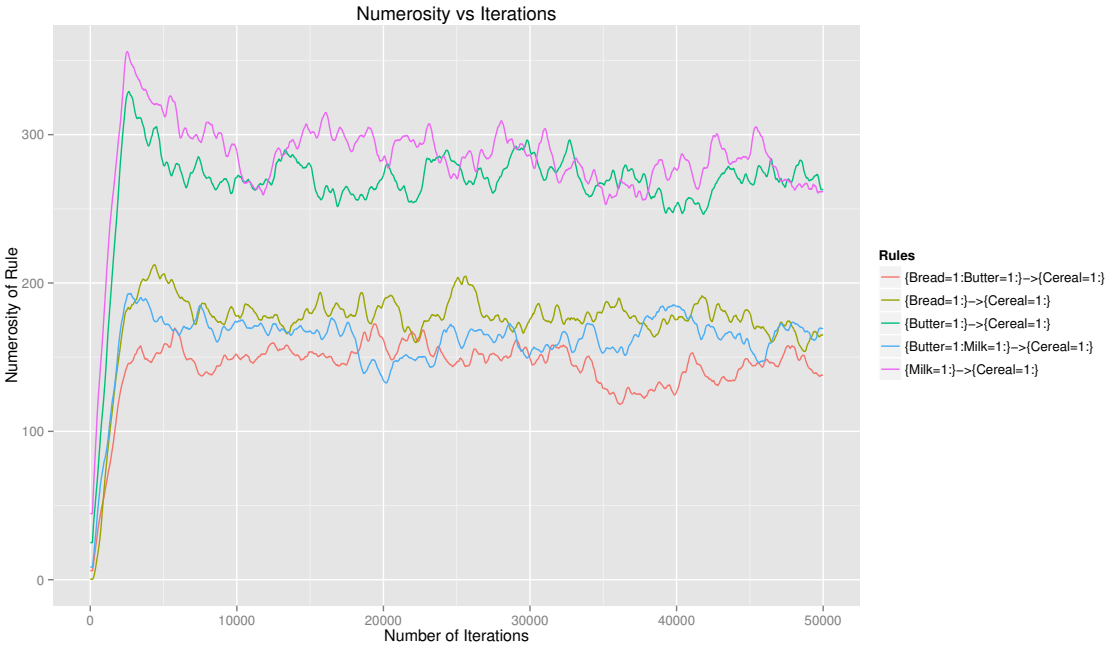


Figure 4.9: Numerosity vs Iterations

Figure 4.10: Numerosity values for rules with Cereal in the consequent, using the CSAR configuration and a population of 5000, removing the GA trigger and using a panmictic GA.

rules with only two attributes were counted, these rules can never be subsumed therefore are the most important to produce for the user if fit enough. Table 4.11 shows the average number of rules with 2 items over 10 runs of the algorithm, it can be seen that there is significantly less rules using a niche GA, whilst the best results are using a panmictic GA and no trigger.

Niche GA	Panmictic GA	Panmictic GA and No Trigger
61.7	111.3	156.4

Table 4.11: Comparing the number of rules with 2 attributes hence non-subsuming, when using differing selection schemes and removing the GA Trigger

4.1.4 Benefits of Overlap Competition

There has been substantial focus in this section on the disadvantages of rules overlapping and being competed out of the population, however XCS has been designed to take advantage of the intrinsic competition within niches. Consequently there are reasons for this and therefore advantages of competition within the algorithm, to provide a thorough analysis of the solutions presented it is also necessary to determine whether there are substantial reasons competition should not be removed

Removing low fitness rules Competition is the core of genetic algorithms and their design around the ideas of evolution and natural selection. Therefore competition is needed to ensure, less fit rules are deleted from the population and allowing fit rules to take over the population and succeed. However due to the nature of association rule mining and the number of rules needed, it is important for more than the maximally fit rules to succeed. Therefore it can be concluded that competitions is

beneficial, however it is important that this competition is managed ensuring fit rules are not deleted as attempted in the fixes presented.

Rule Reduction It has been mentioned numerous times that association rule miners output a large number of rules, therefore it may be advantageous to the user to have a reduced rule set for a more succinct description of the dataset. This can be given by allowing high competition and therefore overlap competition, it is not infeasible for users to only want fittest rule that describes a region. Therefore the ability for users to use a traditional XCS configuration is available, furthermore the results found in this section provide a further understanding into what this means for the output of the algorithm. However it has been found that even the fittest rules can be deleted, if there are other rules within a niche the are of the same fitness.

Implicit Support Pruning Traditional association rule mining methods use supports monotonically decreasing property to prune the search graph, this is not possible while using an online learner due to the limited information available. Additionally a rule can have low support but high confidence, whilst this algorithm takes into consideration support within the fitness function, support also indirectly affects the numerosity of a rule due to the ability to be in a match set and therefore chosen for reproduction.

4.1.5 Overlapping Rule Conclusion

Determining whether overlapping rules were creating harmful competition was a major research question and contribution of this project. It has been determined and therefore answered one of the research questions, that overlapping rules have a detrimental affect on mining association rules. This is not only a contribution to mining association rules using a LCS inspired algorithms, but also provides additional support that this problem could be present elsewhere in the wealth of LCS applications.

Additionally not only has a conclusion been drawn in that there is a problem when rules overlap, causing them to loose numerosity and in extreme cases be removed from the population, but there has be a focus on three causes each of which has been proven to affect this problem. This understanding of the causes has enabled the ability to provide solutions for the problem, which have been presented to positively contribute towards removing harmful competition between overlapping rules when using a panmictic GA and removing the GA trigger. Additionally although there are advantages as suggested above, competition is still present through the use of a panmictic GA, hence these advantages should still be present in the improved algorithm.

4.2 Investigating the Online Learning Robustness

4.2.1 Description

The ability to mine transactions online was presented as one of the main benefits of CSAR in Orriols-Puig and Casillas (2010), however it was noted that there were no results to show the robustness of the algorithm adapting to a changing distribution in the dataset. In fact the future work proposed in Orriols-Puig and Casillas (2010) suggested tests needed to be carried out to verify, that the

algorithm could in fact effectively mine rules online. It is noted that there are some changes to the CSAR algorithm within this thesis, however the main mechanisms in terms of using an LCS inspired algorithm are similar. Therefore it can be presented that these results are a contribution to not only this algorithm, but an extension of the results provided in (Orriols-Puig and Casillas, 2010).

From the results presented above on overlapping rules, it becomes apparent that there is the possibility that when the dataset starts changing new rules could overlap with old. This could potentially lead to new or old rules being deleted. However this behaviour maybe desirable for the user, depending upon how old a rule is, it could not represent the current state of the transactions being given to the system. Therefore guidelines need to be set upon how this online robustness can be evaluated both quantitatively and quantitatively, most research into online learning association rule miners use large frequent itemsets as a base for incoming transactions and graph processing to identify frequent itemsets and therefore association rules, this was used in Hidber (1999).

4.2.2 Experimentation

In this section the robustness of the online XCSAR will be evaluated, this will be carried out using two datasets with the same attributes ensuring that they have different support and confidence values for rules, along with different rules. This will allow testing to ensure fitness is updated, when the distribution changes along with new rules being found and rules becoming less fit and deleted. The two datasets will be provided to the algorithm, which will swap datasets after half of the number of generations have passed

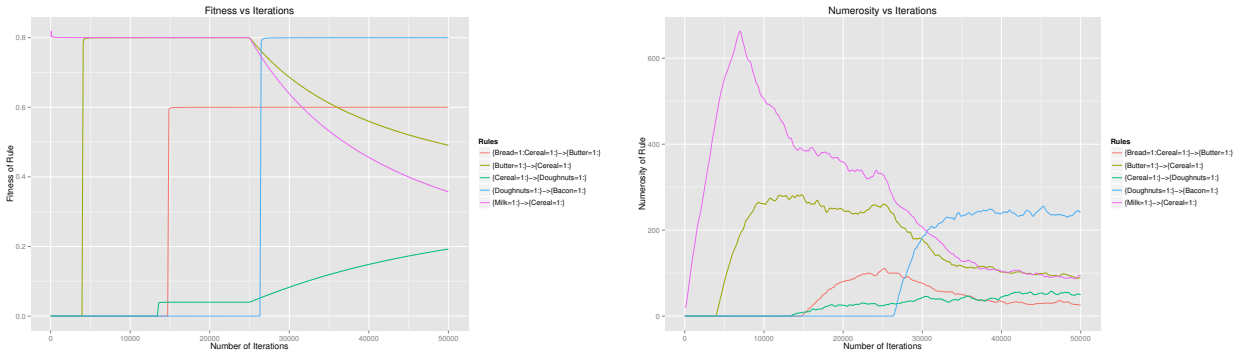
A similar dataset used in the overlapping section was used and a second dataset created, changing the distribution of itemsets. These datasets can be viewed in Appendix B. There is 3 scenarios from these datasets that can be evaluated:

1. Rules not present in the first dataset but present in the second.
2. Rules included in the first dataset, but their support and confidence values increase in the second.
3. Rules with high support and confidence in the first, but decrease in the second.

The algorithm was ran using the two datasets, over 50000 generations and a population of 1000 therefore the algorithm switched to accepting transactions from the second dataset after 25000 generations. Figures 4.11a and 4.11b shows at first glance that the fitness and numerosity values decay and increase according to the change in fitness and support values shown in Table 4.12. However a closer look shows that perhaps the values are not as expected, this can happen in scenario 2 and 3 from above. Rules that are found within both datasets but have their fitness values change, can provide unintuitive results at the end of the learning process as fitness values from the first dataset can heavily affect fitness calculations in the second. This means that very fit rules within the first dataset, can survive using this fitness. This fitness would be correct if there was not an obvious drift in the dataset or simulated drift in this case, however due intuition that the algorithm would adapt to the changing distributions within an online environment this effect will often be unwanted. The user may deem that they would like rules to decay quicker, so rules do not live off previous high fitness for long into the algorithm. Examples of these types of rules is $\{Cereal = 1\} \rightarrow \{Doughnuts = 1\}$,

where both its support and confidence increase in the second dataset however fitness does not increase to its fitness in dataset 2 of 0.4 due to it being impeded by its values from dataset 1. The opposite can be seen for rule $\{Milk = 1\} \rightarrow \{Cereal = 1\}$.

A problem that is definitely unwanted, is when a rule antecedent is not matched at all in the second dataset, this would cause the rule to never be present in a match set and therefore never have their fitness updated. This would cause a rule from the first dataset having a persistent fitness value, which is not representative of its actual fitness value. Additionally if a panmictic GA, was chosen to be used this could cause this rule to be unfairly selected due to this lack of fitness update, although within a niche GA it would not be chosen it can easily survive through the lack of deletion due to its high fitness. An example of this type of rule is $\{Bread = 1, Cereal = 1\} \rightarrow \{Butter = 1\}$.



(a) Fitness vs Iterations

(b) Numerosity vs Iterations

Figure 4.11: Fitness and numerosity values for a variety of rules, that distributions change when changing the datasets

Rule	**Dataset 1**			**Dataset 2**		
	Supp.	Conf.	Antecedent	Supp.	Conf.	Antecedent
$\{Bread = 1, Cereal = 1\} \rightarrow \{Butter = 1\}$	0.6	1	0.6	0	0	0
$\{Butter = 1\} \rightarrow \{Cereal = 1\}$	0.8	1	0.8	0.4	0.67	0.6
$\{Cereal = 1\} \rightarrow \{Doughnuts = 1\}$	0.2	0.2	1	0.4	1	0.4
$\{Doughnuts = 1\} \rightarrow \{Bacon = 1\}$	0	0	0.2	0.8	1	0.8
$\{Milk = 1\} \rightarrow \{Cereal = 1\}$	0.8	1	0.8	0.2	0.33	0.6

Table 4.12: Showing the values for support, confidence and support of the antecedent of the rule for each of the datasets

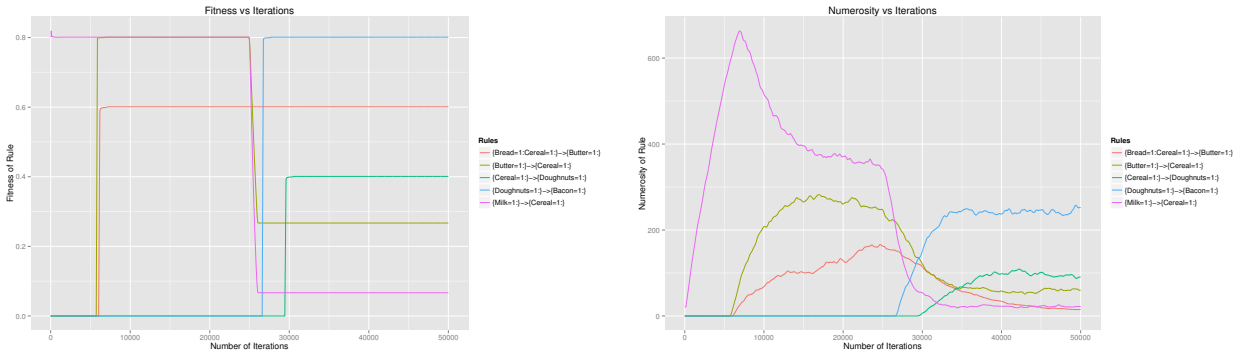
4.2.3 Improving Online Performance

As shown in the previous section, there are two problems one of which should be decided by the user upon how much influence past fitness have on the future of that rule. Additionally it was also found that rules, which match and are found at some point during the algorithm and never match an input again, never have their fitness updated. Along with this it can be seen that this problem, could also lead to another where a rule could be found and then not matched for many generations then start matching many inputs. This large gap would heavily affect the rules fitness due the fitness function using time as a denominator. This problem can be seen as a mixture of both problems, therefore any solution would ideally provide a partial solution to this new scenario.

Calculate fitness over moving window

To enable users to determine how much past fitness affects the current fitness of a rule, it is necessary to calculate fitness over a moving window, giving the user the ability to set this size of this moving window. This therefore removes the problem of rules, living off previous success or being impeded by fitness that was previously poor. Additionally this would also fix the problem stated above, where rules could be found and then have a large amount of generations not matching an input. The results of this change can be seen in Figure 4.12, where rules that were previously obtaining an average fitness over both datasets now has a fitness that is more upto date with current input.

However although calculating fitness over a window of generations greatly improves the online learning robustness of this algorithms, there are computational costs both in terms of time and memory. Due to need to store the iterations that the rule and both its antecedent and consequent match, there is a memory and time cost of $\mathrm{O}(n)$, where n is the size of the window. However this is considered necessary to allow the user the ability to control, when a rule should decay in fitness. This window could also be set as a timeframe, allowing a description of the dataset over a time period. However this approach would need a more efficient way of storing, when a rule matches.



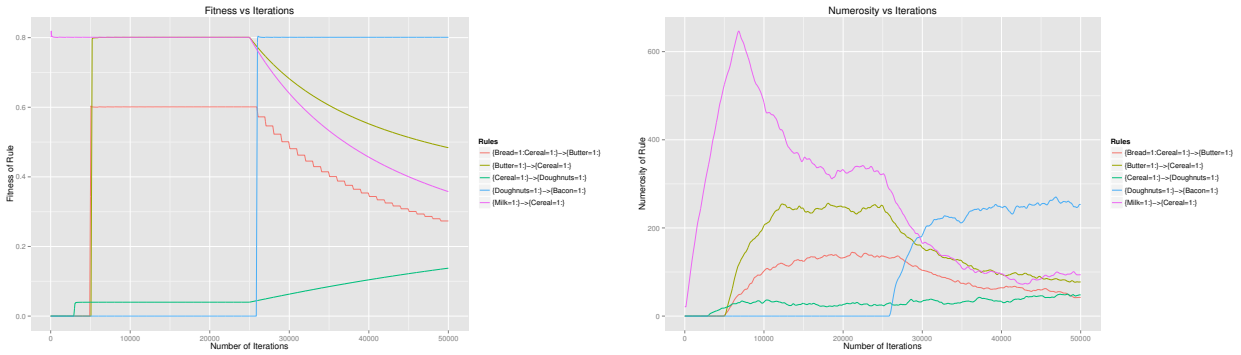
(a) Fitness vs Iterations

(b) Numerosity vs Iterations

Figure 4.12: Updating the fitness over a moving window of 1000 generations, it can be seen that rules that previously were living off high fitness from dataset 1, have had their fitness reduced to the correct fitness for dataset 2.

Updating fitness of whole population

Rules that have not matched for many generations will not have their fitness updated, therefore the ability was provided to update the entire population at intervals of the generations. The results of this can be seen in Figure 4.13, where fitness of the entire population is updated every 1000 generation. This has provided a very effective solution for updating unmatched rules as can be seen with rule $\{Bread = 1, Cereal = 1\} \rightarrow \{Butter = 1\}$, however it was most effective when both methods were used in conjunction. This ensures that when a rules become completely obsolete as $\{Bread = 1, Cereal = 1\} \rightarrow \{Butter = 1\}$, its fitness will be reduced to 0 due to the calculation of fitness over a window.



(a) Fitness vs Iterations

(b) Numerosity vs Iterations

Figure 4.13: Updating the fitness of the whole population every 1000 generations, ensuring rules that have not matched any input have their fitness updated.

4.2.4 Online Learning Robustness Conclusion

In conclusion the online robustness of this algorithm has been evaluated, it was initially found that although the algorithm provided some change when the distribution of rules fitness changed within the input. There were problems especially when rules, never matched another input. The solutions presented although relatively simple have greatly increased the algorithms ability to mine rules online, which was presented as one of the main advantages for using an LCS type algorithm both in this project and (Orriols-Puig and Casillas, 2010). Therefore this improvement can be deemed a large success in improving LCS type algorithms ability to mine associations rules online, which will no doubt become important due to the velocity of new data presented on the internet and in many scenarios association rule miners thrive. However in the future a more robust evaluation will be needed, testing the algorithms ability of true streams of data. This is considered as future work due to the significant work this will entail, which would most likely be enough to carry out a future project of similar size to this.

4.3 Chapter Conclusion

This chapter has investigated the expected problems in XCS recently found, there has been a significant evidence presented that this problem is present when XCS is applied to association rules. Therefore this chapter has answered one of the questions presented to the project, in that overlapping competition can have a significant impact on the algorithms ability to provide fit rules. The problem was found to be mainly caused by the fitness sharing mechanisms in XCS, however the niche GA and trigger also have negative impacts. This allowed changes to be made to enhance the performance of this algorithm, which were proved to provide a significance performance increase against the traditional XCS configurations.

Additionally there has been a short evaluation of the online learning robustness of XCSAR, which was found to have problems specifically in the fitness updates of the algorithm. Therefore solutions were presented to improve upon this, whilst setting out work for a future project further evaluating XCSAR online robustness.

The next section will use these improvements made from this chapter to evaluate the performance

of XCSAR against Apriori, QuantMiner and clustering determining XCSAR ability to mine fit rules whilst evaluating the rules interestingness.

Chapter 5

Evaluation of Algorithm

This chapter will delve into the evaluation of XCSAR and its effectiveness of evolving association rules, with categorical and continuous attributes. This evaluation will include determining whether XCSAR can produce interesting rules, along with comparing the output against well founded algorithms such as Apriori for categorical datasets and QuantMiner for continuous/quantitative. It should be noted that quantitative attributes is a term used by association rule miners, due to the link to purchasing quantities, however the terms quantitative and continuous interchangeable when speaking in more general terms.

5.1 Evaluation Methodology

The objectives for this evaluation are 3 fold, *1*) ensuring XCSAR can actually produce association rules inline with traditional algorithms, *2*) exploring the quantitative rules XCSAR produces and whether they are meaningful and interesting and *3*) determining whether the additions to the algorithm such as the maxInterval setting are effective. .

To allow effective evaluation of XCSAR this section will provide the methodology to determine the effectiveness of the algorithm, along with identifying how XCSAR can be compared against existing algorithms. It is known that Genetic Algorithms as implemented in XCSAR are heuristic in nature, therefore do not always lead to the most optimal solution which in terms of association rules and this project is classed, as providing all maximally fit rules as an exhaustive approach such as Apriori. Despite this acknowledgement of Genetic algorithms not always providing the most optimal solution, there has been justification provided for Genetic Methods in Chapter 2, highlighting the ability to mine intervals and rules at the same time.

Firstly evaluations will be made on categorical datasets, it is noted that much of the analysis in the previous chapter was on smaller datasets. Therefore it is paramount that this evaluation be undertaken on larger datasets, found from the UCI repository. One of these datasets has 30000 instances and another 23 attributes, which as discussed exponentially increases the search space allowing a thorough evaluation. The categorical datasets will be mined by Apriori as discussed. Whilst the objective is not to produce all rules found by Apriori, it is paramount that the XCSAR algorithm is able to produce to highly fit rules, the criteria for XCSAR being deemed a success while mining categorical datasets.

Evaluating XCSAR's ability to mine quantitative/continuous attributes is not as straight forward as categorical, due to the lack of well founded and highly regarded algorithm as Apriori. Therefore this evaluation will have to include algorithms from other areas of data mining, in particular the kmeans clustering algorithm. Additionally it is also possible to plot pairwise attributes in a dataset, giving the ability to visually evaluate the rules produced.

As mentioned there is not an algorithm that is as highly regarded as Apriori, however Quant-

Miner was developed to use Apriori rules as templates to mine intervals with a Genetic algorithm. Therefore the output of QuantMiner will be used to compare the fitness of the rules produced by both algorithms, similar to the process of using Apriori with categorical attributes.

Therefore the objectives are for XCSAR to produce rules that can describe clusters in a datasets, along with producing rules to a similar maximal fitness as QuantMiner. This will show that XSCAR can produce not only meaningful and interesting rules, but is also focusing on rules with high fitness.

The results from the previous section determined that significant problems can occur with overlapping rules becoming extinct when there is substantial competition. Therefore in this evaluation the algorithm will use a panmictic GA and remove the GA Trigger, in an attempt to present the correct rules to the user, even if rules overlap. The algorithm will be ran for 100000 generation, with a population size of 5000. As previously found when XCSAR operates at full capacity, this can have a detrimental affect on the rules produced. However it is thought that the fixes presented should reduce this problem and maximally fit rules be found.

5.2 Evaluation of Categorical Rule Mining

The main alternative for mining datasets with categorical attributes is Apriori, therefore it is paramount that XCSAR is evaluated against Apriori. This will determine whether XCSAR has the ability to mine datasets with categorical attributes.

This section will focus on 3 datasets with categorical attributes comparing the output of both algorithms, the details of these datasets can be viewed in Table 5.1.

Dataset	No. Instances	No. Attributes
Adult	30162	8
Mushroom	8124	23
Zoo	101	17

Table 5.1: Information on the datasets to evaluate the categorical capabilities of XCSAR.

It can be seen from Table 5.2 that XCSAR is able to produce very similar output to Apriori at high fitness levels, which has been suggested as desirable. At lower fitness levels XCSAR is unable to produce as many rules, however this is due to the population limits and it is expected that all rules would not be produced by XCSAR. However due to this ability to focus on maximally fit rules, the user is only presented with rules that are highly fit and therefore interesting. This can be seen as an advantage due to the often extremely large populations of rules produced by exhaustive methods, hindering the ability to easily analyse the rules produced. This affect can be simulated in Apriori by setting a minimum support threshold, which is not needed within XCSAR due to the fitness heuristic guiding the GA, allowing less input from the user and removing the experimentation needed to choose the correct minimum support.

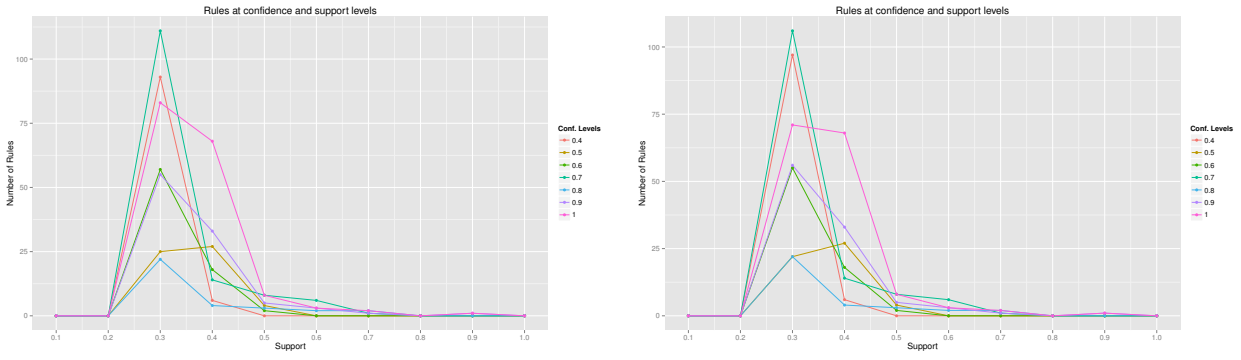
Table 5.2 only shows the rules that XCSAR produces which overlap with the Apriori output, however while the fitness for the maximally fit rules was correct within 0.05, less fit rules are often found later and can still be deleted and rediscovered even with the improvements analysed in the previous chapter. Therefore as the support and confidence values are estimated online not all values are accurate, therefore rules with experience less than 5000 were not used, in an attempt to have

a higher confidence in the support and confidence values of those rules. Additionally results for a typical XCS implementation are provided, this shows that the changes made in the previous chapter are paramount. This also answers a question presented to this project in that; is an LCS algorithm or XCS more specifically suitable for mining association rules, it can be seen that the answers to this is no. However if changes are made it can produce very good results, even when compared to an exhaustive deterministic method.

Fitness	**Adult**			**Mushroom**			**Zoo**		
	Apriori	XCSAR	XCS	Apriori	XCSAR	XCS	Apriori	XCSAR	XCS
0.1	558	224	48	0	0	0	0	0	0
0.2	514	351	66	0	0	0	126	7	4
0.3	160	153	22	1541	282	5	1751	232	25
0.4	109	109	21	1917	488	4	2248	736	22
0.5	14	14	12	1789	570	4	1322	705	3
0.6	7	7	4	312	227	0	267	240	1
0.7	1	1	1	92	90	2	45	45	3
0.8	2	2	0	56	53	1	5	5	1
0.9	0	0	0	47	47	3	0	0	0
1	0	0	0	13	13	0	0	0	0

Table 5.2: Number of rules at each fitness level for each categorical dataset, this shows that XCSAR is able to produce the maximally fit rules within a dataset. Additionally it can be seen that the changes and analysis produced in the previous chapter are paramount to the success of XCSAR and that a standard implementation of XCS is in no way suitable for mining association rules.

Figure 5.1 shows that XCSAR can produce a very similar output to Apriori when there are a relatively small number of rules found by Apriori, it can be seen that often at lower support and confidence levels Apriori produces slightly more rules. However at the confidence level 0.4 and support 0.3, XCSAR produces slightly more rules. This is due to the estimation of support and confidence being wrong, which is therefore a drawback mining rules online and not traversing the dataset to gain an accurate values. This is seen as a negligible problem due to the advantages bought through mining online and the accuracy of highly fit rules.

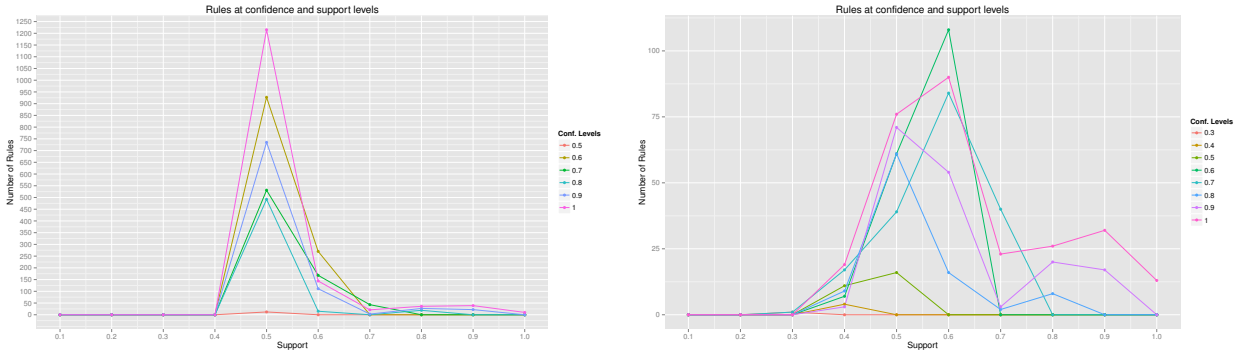


(a) Number of Rules vs Fitness for the Apriori algorithm

(b) Number of Rules vs Fitness for the XCSAR algorithm

Figure 5.1: Number of rules at specific support and confidence levels for the Adult dataset, where confidence is depicted by the colour of the line.

Figure 5.2 shows the number of rules at specified support and confidence values in the mushroom. The mushroom dataset has many more rules produced by Apriori, than is capable of fitting in a population limit. Therefore you would expect that this would cause problems with maintaining fit rules due to the population becoming full. However although many rules are not present, it can be seen that rules with high confidence and support are given as output, even though there are many rules found with low confidence, which would fill the population.

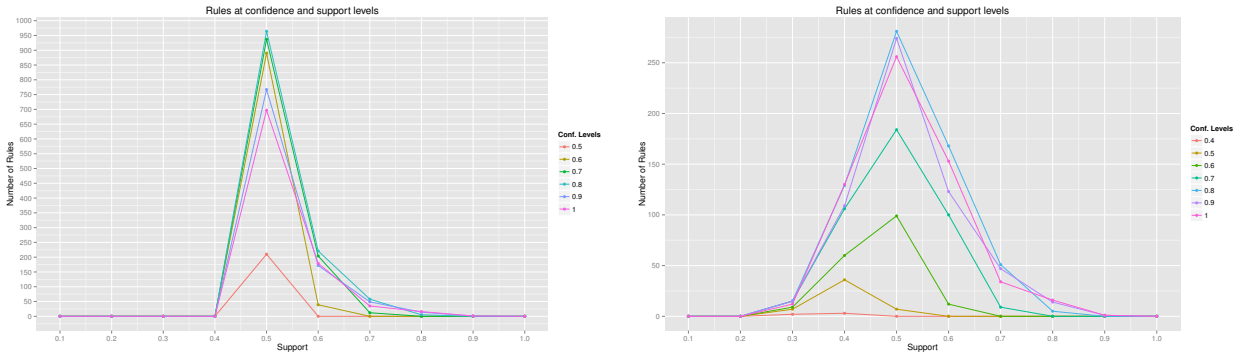


(a) Number of Rules vs Fitness for the Apriori algorithm

(b) Number of Rules vs Fitness for the XCSAR algorithm

Figure 5.2: Number of rules at specific support and confidence levels for the Mushroom dataset, where confidence is depicted by the colour of the line.

Figure 5.3, shows much the same as the previous two datasets in that XCSAR although many less rules are found using Apriori, XCSAR focuses on the same region of rules.



(a) Number of Rules vs Fitness for the Apriori algorithm

(b) Number of Rules vs Fitness for the XCSAR algorithm

Figure 5.3: Number of rules at specific support and confidence levels for the Zoo dataset, where confidence is depicted by the colour of the line.

5.2.1 Categorical Mining Conclusion

In conclusion it has been found that XCSAR performs very well against Apriori and even though Apriori is an exhaustive method, XCSAR has the ability to find many of the rules found by Apriori. This is especially true for the maximally fit rules, where XCSAR produced all the same rules as Apriori. This conclusion not only presents a success for XCSAR when mining datasets with only categorical attributes present, but also allows the mining of datasets with varying attribute types.

Throughout the next section data sets with continuous or quantitative attributes will be evaluated, leading to the evaluation of datasets with both types of attribute. The results have also improved upon the limited results produced in Orriols-Puig and Casillas (2010), where even though similar parameters were used in terms of population size more fit rules were produced by XCSAR. Additionally as XCSAR can mine multistep rules of which CSAR could not, this further improves the output of maximally fits rules, allowing a rule such as $\{Cereal\} \rightarrow \{Milk, Butter, Bread\}$ highlighting the extremely predictive nature of the *Cereal* attribute. This predictive nature as discussed could be used in the first steps of learning about a dataset, instead of using traditional tools such as correlation which is often limited to pairs of attributes.

5.3 Evaluation of Quantitative Rule Mining

Whilst reviewing Orriols-Puig and Casillas (2010) in Chapter 2, it was suggested that the evaluation of the CSAR algorithms ability to mine quantitative association rules was not thorough enough. Additionally, there was no comparison against other algorithms or validation of rules, CSAR was only deemed a success based on the support and confidence of the rules. Therefore this section will evaluate the effectiveness of this algorithm against other quantitative association rule miners. Additionally it is necessary to evaluate the rules produced and whether they are meaningful, association rules should describe groups of customers in the case of shopping basket data or instances more generally. It can therefore be seen that groups of data can also be described by clustering, hence where visualisation is possible; this section will explore XCSARs ability to provide clusters within a dataset. This will produce a robust evaluation into the rules present within the dataset given.

5.3.1 Comparing against Clustering

While it can be seen intuitive to compare this algorithm with clustering, the limitation of both this algorithm and clustering have to be taken into consideration. Due to the need for the maxInterval parameter and the lack of this parameter in any clustering algorithm, the output may not have a one-to-one relation. However it is hopeful that rules should easily describe the most densely populated regions, which are easily distinguishable in 2D and describe similar clusters as the kmeans algorithm in higher dimensions.

This section will focus on the Iris dataset and the Adult using only the continuous attributes in both, firstly exploring the pairwise plots of the attributes and then further evaluating rules in higher dimensions using the k-means clustering algorithm.

Iris Dataset

The pairwise plots of the iris dataset can be seen in Figure 5.4, it can be seen that there are distinguishable clusters that XCSAR should provide rules to describe. Additionally there are multiple clusters within some of the plots, which will determine whether XCSAR can improve upon QuantMiner which can only describe one relation for each set of attributes.

Table 5.3 shows the rules given for each of the pairwise plots previously shown, it can be seen that the rules describe the clusters very accurately in that the most densely populated areas are covered by the rules given. Additionally it can be seen that unlike QuantMiner, multiple rules can be given

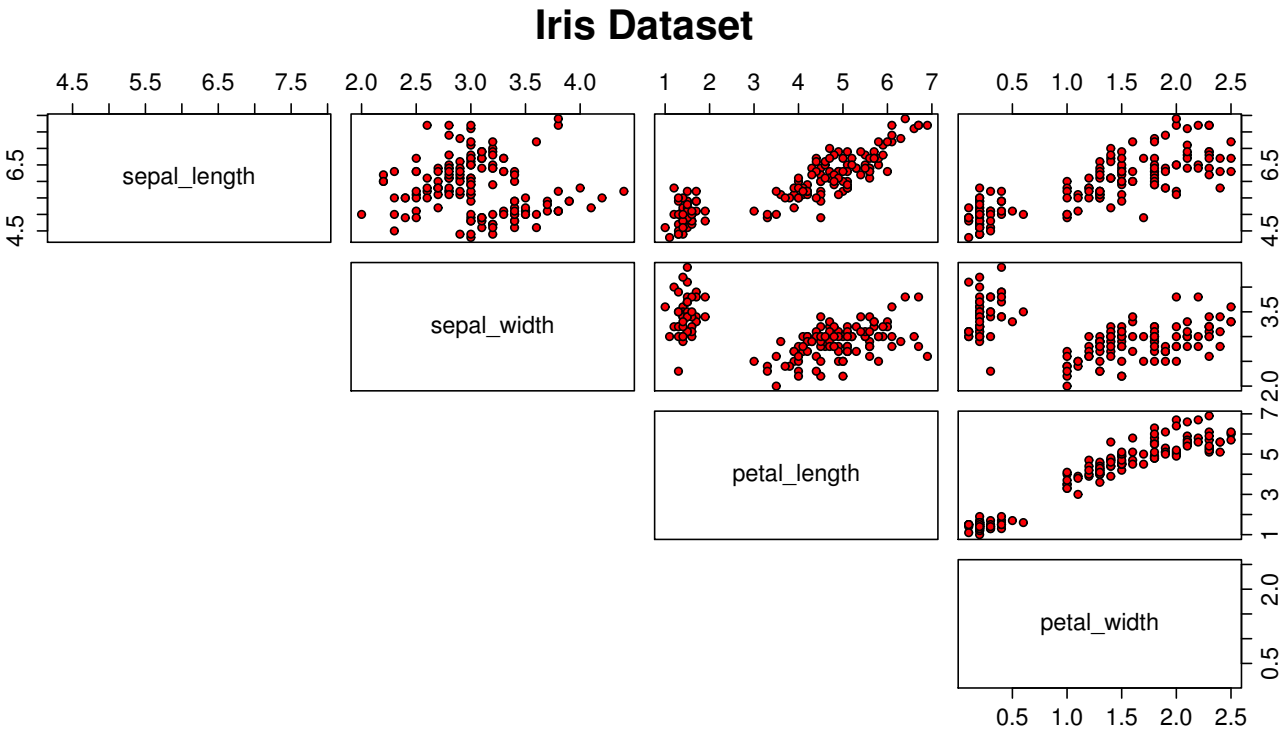


Figure 5.4: Pairwise plot of the continuous attributes in the Iris Dataset

for the same set of attributes. This gives the ability to describe multiple clusters within that set of attributes. This improvement upon QuantMiner was suggested as future work in that finding logical OR rules, would be advantageous and a future improvement. The rules displayed can be combined to fully fulfill this future work and producing rules with ORs. Table 5.4 shows the ability to combine two rules, although this was not carried out in XCSAR as it is thought to hinder the readability of the rule.

Pairs	**Rules**			
	Rule	Supp.	Conf.	Fitness
sepal_length vs sepal_width	$\{sepal\_length = [5.5, 6.9]\} \rightarrow \{sepal\_width = [2.5, 3.3]\}$	0.471	0.831	0.391
sepal_length vs petal_length	$\{sepal\_length = [5.7, 7.3]\} \rightarrow \{petal\_length = [3.0, 6.4]\}$	0.5	0.96	0.48
	$\{petal\_length = [1.0, 1.9]\} \rightarrow \{sepal\_length = [4.6, 6.1]\}$	0.3	0.9	0.27
sepal_length vs petal_width	$\{petal\_width = [0.1, 0.2]\} \rightarrow \{sepal\_length = [4.4, 5.8]\}$	0.18	0.84	0.15
	$\{petal\_width = [1.0, 2.2]\} \rightarrow \{sepal\_length = [5.7, 7.3]\}$	0.47	0.74	0.35
sepal_width vs petal_length	$\{petal\_length = [1.0, 1.9]\} \rightarrow \{sepal\_width = [2.9, 3.6]\}$	0.24	0.71	0.17
	$\{petal\_length = [3.0, 6.4]\} \rightarrow \{sepal\_width = [2.5, 3.3]\}$	0.54	0.84	0.46
sepal_width vs petal_width	$\{petal\_width = [0.1, 0.2]\} \rightarrow \{sepal\_width = [2.9, 3.6]\}$	0.17	0.76	0.13
	$\{petal\_width = [1.0, 2.5]\} \rightarrow \{sepal\_width = [2.4, 3.2]\}$	0.55	0.84	0.46
petal_length vs petal_width	$\{petal\_width = [1.0, 2.2]\} \rightarrow \{petal\_length = [3.0, 6.4]\}$	0.61	0.96	0.58
	$\{petal\_width = [0.1, 0.4]\} \rightarrow \{petal\_length = [1.0, 1.8]\}$	0.22	0.7	0.15

Table 5.3: Output of the reduced ruleset for rules describing the pairwise associations

Pair	XCSAR Output	Combined Rule
petal_length vs petal_width	$\{petal\_width = [1.0, 2.2]\} \rightarrow \{petal\_length = [3.0, 6.4]\}$ $\{petal\_width = [0.1, 0.4]\} \rightarrow \{petal\_length = [1.0, 1.8]\}$	$\{petal\_width = [0.1, 0.4]\ OR\ [1.0, 2.2]\} \rightarrow \{petal\_length = [1.0, 1.8]\ OR\ [3.0, 6.4]\}$

Table 5.4: Combining rules as was present in Salleb-Aouissi et al. (2007) future work section.

It has been shown that XCSAR can produce rules that describes clusters in 2D, however this can easily be done visually using the pairwise plots, a more useful tool would be the ability to describe rules in a higher dimensions. Therefore the top two rules describing two sets of three attributes was taken from the reduced ruleset produced by XCSAR. The k-means clustering algorithm was then ran on both sets of three attributes, with the number of clusters set to two, allowing easier evaluation.

Table 5.5 shows the top 2 rules for each set of 3 attributes, it can be seen that these rules have very high confidence showing the strength of the association found. Table 5.6 shows the clusters that was produced by k-means, giving the minimum and maximum for each attribute, hence depicting the cluster boundaries. These tables show that XCSAR is very good at describing clusters in this higher dimension, although the boundaries are not exactly the same, the difference is minor and is generally due to the maximum interval restriction.

Rule No.	Rule	Supp.	Conf.	Fitness
1	$\{petal\_length = [3.0, 6.4] sepal\_width = [2.5, 3.3]\} \rightarrow \{petal\_width = [1.0, 2.2]\}$	0.54	0.99	0.54
2	$\{petal\_length = [1.5, 4.0] sepal\_width = [2.4, 3.2]\} \rightarrow \{petal\_width = [0.1, 1.4]\}$	0.24	0.8	0.19
3	$\{petal\_width = [1.0, 2.1] sepal\_length = [5.5, 6.9]\} \rightarrow \{petal\_length = [3.3, 6.6]\}$	0.5	1.0	0.5
4	$\{petal\_length = [1.0, 1.9] sepal\_length = [4.6, 6.1]\} \rightarrow \{petal\_width = [0.1, 1.4]\}$	0.3	1.0	0.3

Table 5.5: Rules given by XCSAR that are able to describe clusters in 3 dimensions and are very similar to the output presented by kmeans.

Cluster	sepal_width		petal_length		petal_width	
	Min	Max	Min	Max	Min	Max
1	2.0	3.8	3.3	6.9	1	2.5
2	2.3	4.4	1.0	3.0	0.1	1.1

(a) Cluster boundaries for two clusters using the attributes sepal_width, petal_length and petal_width

Cluster	sepal_length		petal_length		petal_width	
	Min	Max	Min	Max	Min	Max
3	4.9	7.9	3.5	6.9	1	2.5
4	4.3	5.8	1.0	3.5	0.1	1.1

(b) Cluster boundaries for two clusters using the attributes sepal_length, petal_length and petal_width

Table 5.6: Cluster boundaries to compare against two sets of rules

Adult Dataset

The adult dataset is census data, which has attributes regarding a persons financial situation, age and employment information. Figure 5.5 shows two plots for a persons capital loss against age and number of working hours per week. These figures show the necessity to not only focus on one region of a set of attributes, as QuantMiners does. It can be seen that there is a large amount of instances, with capital loss of 0. Therefore QuantMiner produces rules displayed in Table 5.7. These rules although defining a significant cluster within the pair of attributes, they are perhaps not as interesting to a user than the cluster of instances in the centre of both graphs, Table 5.8 shows the rules given by XCSAR. This shows that not only does XCSAR find the significant grouping of instances at capital loss of 0, but also the alternate association depicted by the cluster of instances in the center of the graphs. As said it is thought that this association would be much more interesting to a user than instances at capital loss at 0, which could be due to the lack of information for those instances. Furthermore it could be argued that the data could be preprocessed, if data was deemed missing. However, this author argues that XCSAR can be used before any preprocessing of data and used as a tool to find this missing data, whilst still giving

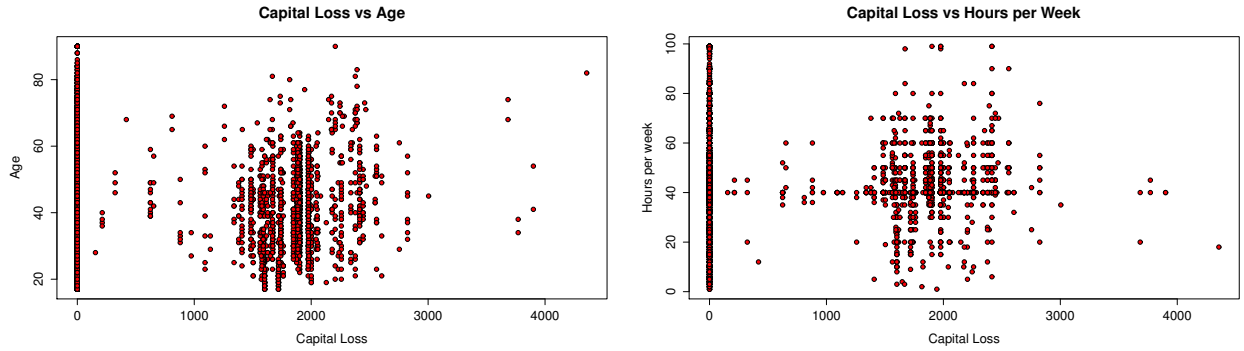
Rule No.	Rule	Supp.	Conf.	Fitness
1	$\{hours.per.week = [40.0, 40.0]\} \rightarrow \{capital.loss = [0.0, 0.0]\}$	0.41	0.96	0.43
2	$\{age = [22.0, 37.0]\} \rightarrow \{capital.loss = [0.0, 0.0]]\}$	0.4	0.96	0.39

Table 5.7: Output from QuantMiner algorithm, describing only one cluster.

Rule No.	Rule	Supp.	Conf.	Fitness
1	$\{hours.per.week = [29.0, 42.0]\} \rightarrow \{capital.loss = [0.0, 0.0]\}$	0.56	0.95	0.53
2	$\{capital.loss = [1500, 2463]\} \rightarrow \{hours.per.week = [34.0, 45.0]\}$	0.12	0.45	0.05
3	$\{capital.loss = [1349, 2023]\} \rightarrow \{age = [24.0, 40.0]\}$	0.15	0.5	0.075
4	$\{age = [27.0, 42.0]\} \rightarrow \{capital.loss = [0.0, 0.0]\}$	0.38	0.95	0.36

Table 5.8: Output from XCSAR for the 3 attributes from the adult dataset

Additionally due to the significance of this group in higher dimensions, QuantMiner produces many rules that only describe this relationship of capital loss at 0. Whereas XCSAR can describe additional clusters in higher dimensions, as seen from the Table 5.9, compared to the one rule given by QuantMiner in Table 5.10. Additionally running kmeans on the attribute capital loss, age and hours per week with two clusters gives similar cluster as given by XCSAR which can be seen in 5.6, where the clusters are depicted by the colours of the instances.



(a) Capital Loss vs Age, showing 2 clusters of data (b) Capital Loss vs Hours per week, showing 2 clusters

Figure 5.5: The plots presented show that although there is significant number of instances where capital loss is 0, there are other clusters that can be potentially more interesting to the user

Rule No.	Rule	Supp.	Conf.	Fitness
1	$\{capital.loss = [0.0, 0.0], hours.per.week = [35.0, 40.0]\} \rightarrow \{age = [20.0, 46.0]\}$	0.37	0.72	0.27
2	$\{capital.loss = [950.0, 3000.0], hours.per.week = [40.0, 45.0]\} \rightarrow \{age = [20.0, 40.0]\}$	0.2	0.67	0.134

Table 5.9: Output from XCSAR for 3 attributes from the adult dataset, describing a two clusters which are similar to the kmeans clustering algorithm of which one is not given by QuantMiner

Rule No.	Rule	Supp.	Conf.	Fitness
1	$\{capital.loss = [0.0, 0.0], hours.per.week = [40.0, 40.0]\} \rightarrow \{age = [23.0, 40.0]\}$	0.23	0.52	0.12

Table 5.10: Output from QuantMiner for the 3 attributes from the adult dataset, this rule only describes one cluster whereas there are clearly two.

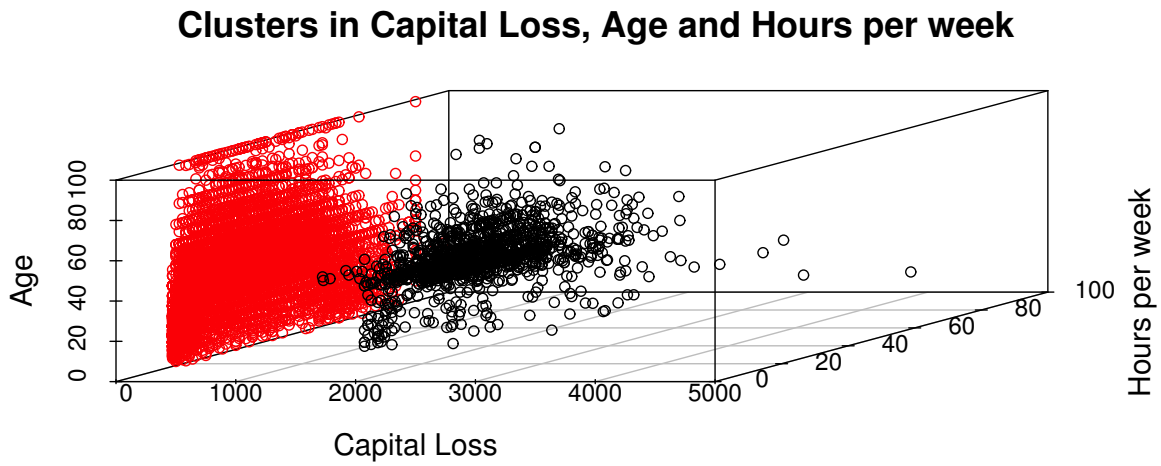


Figure 5.6: 3D plot of Capital Loss, Age and Hours per week show the clusters identified by kmeans depicted by colour.

In conclusion these results suggest that XCSAR can produce rules to describe clusters as expected, this is not only a measure of how meaningful the rules produced by XCSAR are, but it also shows that XCSAR can be used a tool to help understand datasets. There are some obvious advantages of using association rules to describe clusters rather than using traditional method like kmeans, these include; readability and intuition of rules, the number of clusters does not need to be given and associations or clusters can be found using differing amounts of attributes, which would be infeasible with kmeans as all combinations of attributes would need to be explored.

5.3.2 Comparing QuantMiner Output

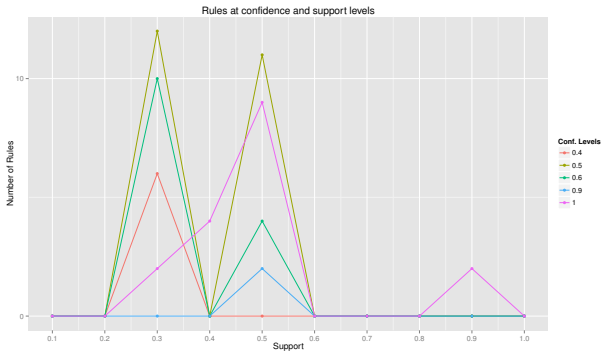
In this section, 5 datasets will be used to compare the output of XCSAR and QuantMiner. However, due to the space requirement only 3 datasets will be evaluated in this section while the final datasets can be found in Appendix C, one of which is the Iris dataset which was evaluated in the previous

section. As found above it is highly unlikely that intervals found by both algorithms will be exactly the same, as with clustering. Additionally, there have been some advances upon QuantMiner within XCSAR, such as the ability to mine multiple rules for the same attributes, giving the ability to describe multiple clusters as shown in the previous section. This improvement means that there are likely to be more rules produced by XCSAR. Therefore the comparison of XCSAR and QuantMiner will focus on; the fitness, support and confidence values for the rules found, this allows a determination of whether XCSAR can mine fit rules.

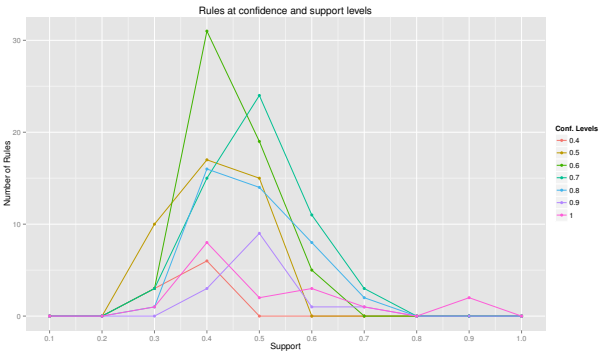
Comparing Rule Output

Adult Dataset The number of rules at each fitness level is depicted in Figure 5.7c, this shows that XCSAR has the ability to mine more and fitter rules than QuantMiner. This was found to be due to XCSAR's ability to focus on small regions with high confidence, due to the algorithms ability to describe multiple clusters as shown in the previous section, in contrast to QuantMiner describing one densely populated area. XCSAR can find many regions, this also allows XCSAR to include all instances in that cluster, whilst still disregarding outliers, however due to QuantMiner only describing one region it classes other clusters as outliers providing no description for these instances.

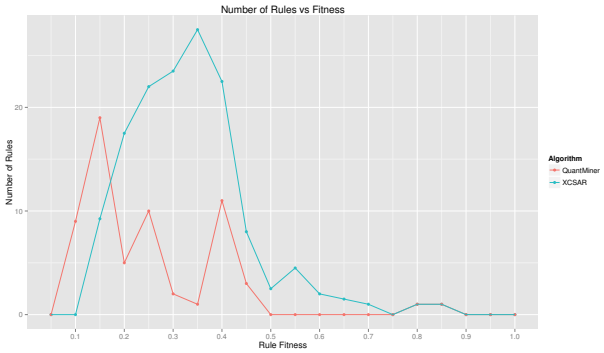
Figure 5.7, further expands on these results showing that XCSAR is able to find more highly confident rules, due to concentration on more than one area of an attributes value range. It is thought that producing these highly confident rules are wanted by the user as they show strong relationships, although it is paramount not to produce rules with high confidence but little support which could be seen as fitting to noise. However, many of the highly confident rules also have high support, therefore showing that the mining of these highly associative smaller clusters, is highly unlikely to be noise.



(a) Number of Rules at specified support and confidence levels for the QuantMiner algorithm



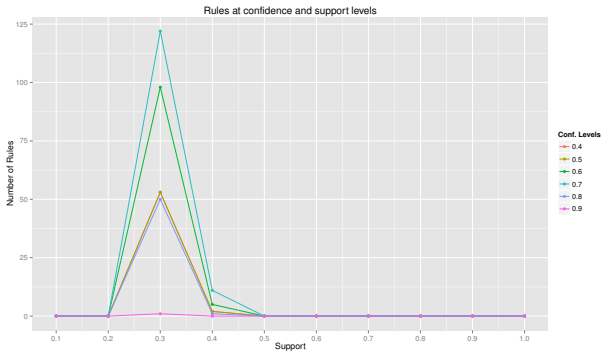
(b) Number of Rules at specified support and confidence levels for the XCSAR algorithm



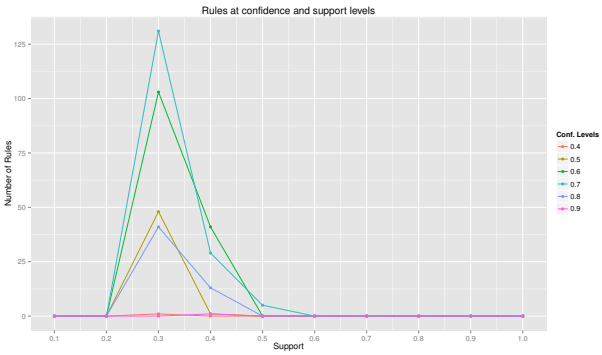
(c) Number of rules at fitness levels for the Adult dataset for both the XCSAR and QuantMiner algorithm, showing that XCSAR produces more fit rules and has the ability to mine the maximally fit rules.

Figure 5.7: Number of rules at support and confidence levels, where confidence is specified by the colour of the line for the Adult dataset. It can be seen that XCSAR again due to the fact that it can mine multiple clusters, produces more rules at higher confidence values. For example QuantMiner produces no rules at a confidence level between 0.6 and 0.9, showing that QuantMiner is missing rules which have a high degree of association.

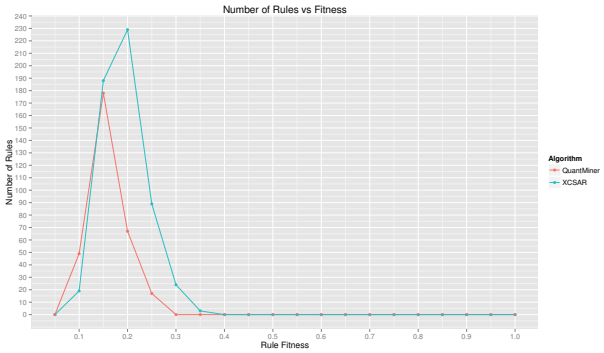
Pima Indians Dataset Figure 5.8, shows the support and confidence values for the Pima Indians dataset. These on first glance show that the algorithms perform similarly on this dataset, however on closer look XCSAR produces more rules at the 0.4 support level. On inspection this was due to XCSAR producing rules with slightly larger intervals, whilst the rules at the lower support values were rules describing the smaller clusters as shown previously in the clustering section. This result is expected and wanted as the reduced population given from the XCSAR algorithm, only gives rules that do not overlap. This shows further that rules are not describing the same region, but alternate regions within the same attribute set. These results can also been seen in Figure 5.8c, where XCSAR produces more fit rules that QuantMiner.



(a) Number of Rules at specified support and confidence levels for the QuantMiner algorithm



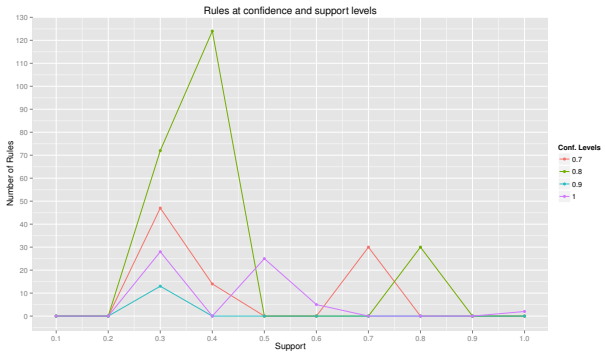
(b) Number of Rules at specified support and confidence levels for the XCSAR algorithm



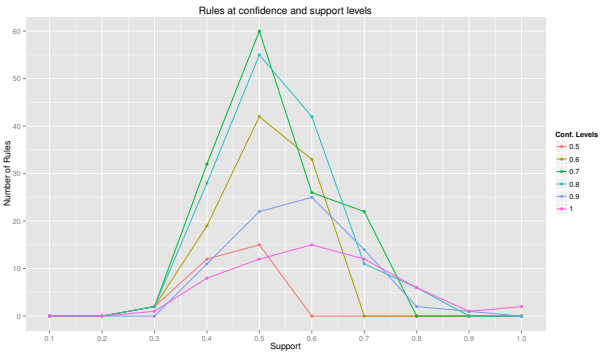
(c) Number of Rules vs Fitness for the XCSAR algorithm

Figure 5.8: Number of rules at support and confidence levels, where confidence is specified by the colour of the line for the Pima Indians dataset.

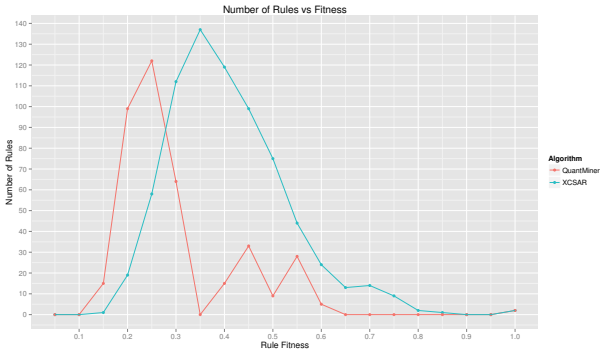
Yeast Dataset Once again Figures 5.9, show XCSAR ability to mine highly fit rules and the slight shift to the right of the graphs shows the slightly larger intervals that XCSAR produces. Additionally as in the previous dataset there are more rules present, due to the ability to describe different clusters. However it can also be seen that there is an overlap of rules with the maximum support and confidence of 1, on closer inspection these rules had exactly the same intervals for both algorithms. This shows that XCSAR has the ability to mine rules, that are of the highest fitness, whilst not expanding the intervals of the rules when it is not needed.



(a) Number of Rules at specified support and confidence levels for the QuantMiner algorithm



(b) Number of Rules at specified support and confidence levels for the XCSAR algorithm



(c) Number of Rules vs Fitness for the XCSAR algorithm

Figure 5.9: Number of rules at support and confidence levels, where confidence is specified by the colour of the line for the Yeast dataset.

Comparing Process

A major comparison between QuantMiner and XCSAR is the process in which the algorithms produce rules with intervals, whereas XCSAR uses a single phase process in that rules and intervals are mined together. QuantMiner first produces a starting point using the output of Apriori, this provides templates for rules, which then gives the ability for the algorithm to focus on those attributes finding the best intervals using a Genetic algorithm. QuantMiner is therefore a two phase process, which also means that it has the overhead of initialising a genetic algorithm for each rule template. This obviously has computational drawbacks and leads to the algorithm taking substantial amounts of time, when more templates are given to the genetic algorithm. This is emphasized due to the first stage being an exhaustive search, which then feeds the less efficient genetic algorithm causing the algorithm to become extremely slow. This can be seen in Table 5.11, where each algorithm was ran on the same dataset 10 times and the average time taken. XCSAR's run time can be seen to be consistent, due to the population size and number of generations staying constant. However, QuantMiner's runtime is highly dependant on the number of rules given by Apriori and therefore the support specified by the user. This reduces usability as not only will setting the support value too low result in many rules for the user, but will also result in considerable run time.

Additionally due to QuantMiners reliance on Apriori being used at the first phase of the algorithm, it is unable to mine rules online and can only work in batch mode. This therefore shows a large

advantage for XCSAR and its ability to mine both rules and intervals at the same time, allowing the algorithm to succeed in coming closer to dealing with the velocity of data, as discussed in the introduction to this dissertation.

	Adult		Iris		Pima Indian		Wine		Yeast	
	XCSAR	Quant	XCSAR	Quant	XCSAR	Quant	XCSAR	Quant	XCSAR	Quant
Avg. Time (s)	612	542	318.8	260	612.2	1006	552	1193.2	898.4	948
P-value	0.007		0.006		4.61E-05		7.43E-05		0.04	

Table 5.11: Average time over 10 runs for QuantMiner and XCSAR, also the p-value for the difference which shows that XCSAR is often fast than QuantMiner, when the Apriori produces more rules for QuantMiners GA to find intervals.

In conclusion it can be seen that XCSAR performs well against QuantMiner, although it is noted that due to the exhaustive first phase of QuantMiner maximally fit rules are always be found. However as discussed this can have detrimental affects to the runtime of the algorithm and its ability to mine online. XCSAR has also introduced advances upon the QuantMiner algorithm which were indicated as future work, notably the ability to mine multiple intervals for sets of attributes. This improvement has been verified to produce interesting results and further aids the users ability to understand the relations within a dataset. Additionally XCSAR generally produces slightly fitter rules, which can be attributed to the shift of the support-confidence graphs to the right, meaning rules have slightly higher support, due to the larger intervals produced by XCSAR. However, work still needs to be carried out to the implementation of XCSAR, whereas QuantMiner provides a graphical user interface to explore rules, XCSAR produces comma separated files. Therefore it is seen as future work to enable the users to explore presented rules easier, to fully take advantage of the advances in the mining process, although this is seen as a secondary aim and in no way takes away the success of the presented algorithm within this project.

5.4 Comparison against CSAR

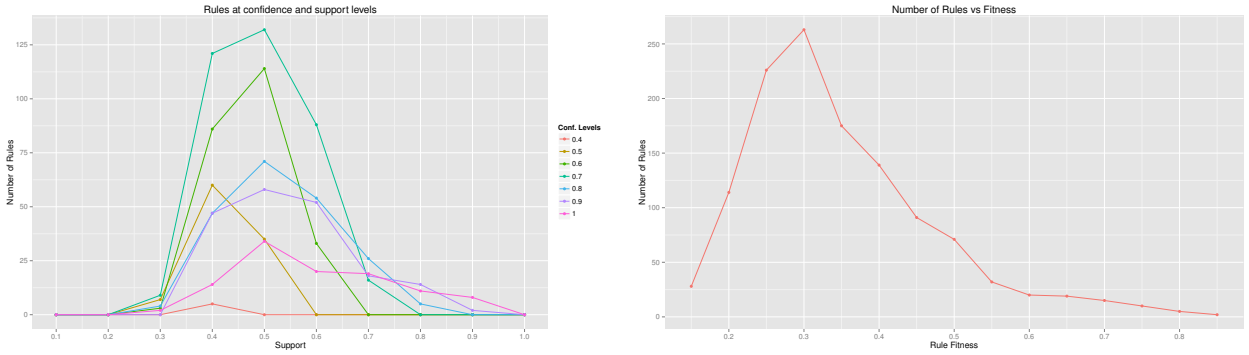
There have been many problems identified from Orriols-Puig and Casillas (2010), which introduced CSAR both in terms of implementation and evaluation. This evaluation has attempted to rectify those problems and the output implementation of this project. Although there is not an available implementation of CSAR, there can be a comparison of the techniques used. There are some very similar elements, however this project has expanded the implementation. XCSAR has the ability to mine multistep association rules, as discussed giving the ability to identify highly predictive attributes. However, more importantly the issue of overlapping has been rectified, giving more confidence that maximally fit rules will be found and presented to the user. Additionally the automatic setting of the maxInterval parameter has greatly increased the flexibility, usability and online robustness of the algorithm. This is due to the need to normalise attributes to be able to manually set a maxInterval of between 0-1, which is not feasible when learning online as the range of values is unknown. Furthermore setting one maxInterval for all attributes greatly reduces the algorithms ability to mine interesting association, as attributes within the datasets evaluated in this chapter, often have very different ranges such as; age and capital loss in the adult dataset, even if these attributes were nor-

malised to have values between 0-1, the spread of the data and therefore the optimal interval would not be the same for these attributes.

5.5 Categorical and Quantitative Mining

XCSAR has been proven to have the ability to mine both categorical and quantitative attributes, therefore this section will present results using both the adult and iris datasets, with all attributes present which include both categorical and quantitative/continuous attributes. Due to the results previously presented, this section will be short as it has been found that both types of attributes can be mined successfully.

Figure 5.10, shows the fitness and support and confidence levels for the the produced by XCSAR. It can be seen that XCSAR is able to focus on regions of high confidence and support. Additionally on closer inspection of the rules produced, there is a very good integration of rules with both continuous and categorical attributes present.

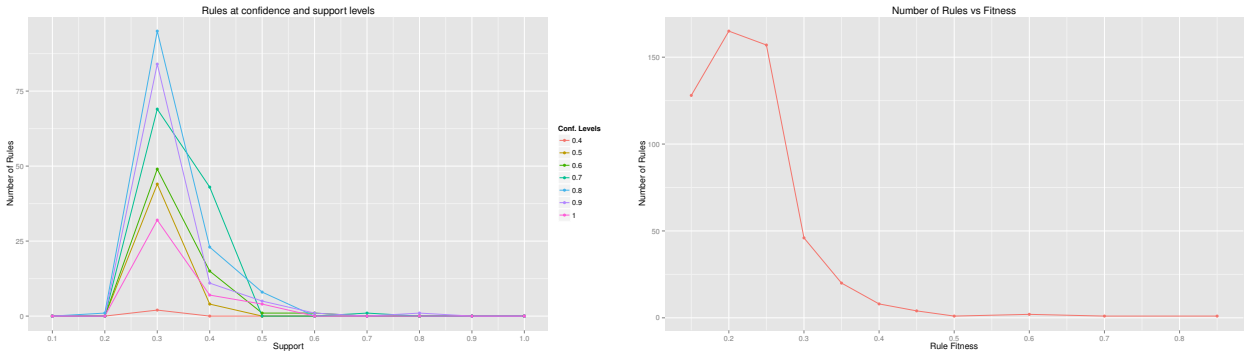


(a) Number of Rules at specified support and confidence levels for the XCSAR algorithm

(b) Number of Rules vs Fitness for the XCSAR algorithm

Figure 5.10: Fitness and support and confidence for rules produced by XCSAR on the Adult dataset with both continuous and categorical attributes

Figure 5.11, shows that XCSAR is also able to produce fit rules from the Iris dataset. Additionally due to the iris dataset being a classification task, rules can be extracted to be used for classification. The rules found with the Class attribute in the consequent, were very similar to the classification rules presented by a classification algorithm such as JRip or J48 trees, this therefore shows that XCSAR can also produce rules that can learn about how to classify, before running a classification algorithm with metrics to determine accuracy rather than association.



(a) Number of Rules at specified support and confidence levels for the XCSAR algorithm

(b) Number of Rules vs Fitness for the XCSAR algorithm

Figure 5.11: Fitness and support and confidence for rules produced by XCSAR on the Adult dataset with both continuous and categorical attributes

5.6 Chapter Conclusion

In conclusion, this chapter has enhanced the CSAR algorithm significantly along with providing a more thorough evaluation, hence determining the success of many of this projects objectives. Notably determining that XCSAR, the algorithm developed in this project has the ability to provide interesting association rules both compared to; previous algorithms and its ability to describe clusters in a dataset. There has been proof that the algorithm, can successfully mine multiple clusters within a set of attributes, an improvement specified in Salleb-Aouissi et al. (2007) as the ability to mine logical ORs of intervals. Along with providing a novel application of using a clustering method to evaluate rules, this allows for improvements to be made to this evaluation in the future, such as automation. In summary the XCSAR algorithm performed very well against both Apriori, clustering and QuantMiner, however in the future it would be interesting to test the algorithms ability to mine much larger streaming datasets.

Chapter 6

Conclusion

This project has presented an implementation of an association rule miner, inspired by Learning Classifier systems with the ability to mine categorical, boolean and continuous attributes online. This chapter will critically evaluate the analysis, implementation and evaluation presented in this report against the objectives set out at the start of this project. Thus providing the ability to conclude as to whether this project has been a success, whilst providing future work to further contribute to the two fields of this project; Association rule mining and Learning Classifier Systems.

6.1 Evaluation

The main and first objective for this project was to develop an association rule mining algorithm, with the ability to generate rules from a multitude of datasets with varying attribute types. There was attempted to use an existing implementation XCSF a function approximator, however it was found that many of the elements were very different from mining association rules. Therefore, the algorithm was written using XCSF as a template for a Learning Classifier System, introducing many of the elements discussed in Chapter 3. It can therefore be concluded, that this objective has not only been met, but the algorithm from this project has thoroughly improved upon previously presented algorithms in terms of QuantMiner and CSAR. These improvements include the ability to mine online, the automatic setting of the maxInterval parameter, the ability to mine multistep rules and multiple clusters within a set of attributes. These improvements have not only enhanced the output presented to the user, but have also increased the usability and flexibility upon mining datasets presented to the algorithm.

However, there are still improvements to be made; one of which is the ability to mine rules with temporal attributes. Initially this was an objective of this project; however, this had to be set as non-essential when the problems with overlapping were found to be more detrimental than originally thought. Therefore, this improvement can be thought of as future work. However, the improvements listed above and the algorithms success on mining alternative types of attributes more than make up for the lack of ability to mine temporal attributes. Additionally, due to the modular design of the algorithm, the introduction of new types of attributes should be relatively straightforward.

The huge amounts of rules produced by association rule miners is usually detrimental to the users ability to digest information that the rules hold; therefore, there was an attempt to produce a reduced rule set. However, there needs to be further work to enable further reduction and therefore comprehension of rules. This comprehension could also be aided by the introduction of a Graphical User Interface for the exploration of presented rules; this would greatly improve the usefulness of the algorithm. Additionally further metrics for association rule mining could be explored,;there has been an attempt to do this with the use of XCS fitness sharing, CSAR's combination of support and confidence and pure support and confidence. It was not thought as feasible to fully analyse

different metrics due to the wealth and analysis provided in Tan et al. (2004) and the favorable results produced by using a combination of support and confidence.

Finally, although XCSAR has found to be efficient in terms of speed compared to QuantMiner, it is not comparable to Apriori in terms of both time and its deterministic nature, due to the well founded graph searching methods of Apriori. However, due to Apriori's inability to search the larger search space of continuous attributes, this can be seen as acceptable in light of the advantages of using a heuristic search on a large search space.

There has been an extensive investigation into the overlapping problem, which initially was not thought to be as extensive as found. However, through the use of a small dataset to understand the complex mechanisms and inherent stochastic nature of genetic algorithms, it was found that three components were contributing to the overlapping problem. It was proven that there were significant effects on the ability of the algorithm to produce rules that overlapped; this problem was paramount to tackle due to the removal of non-subsuming rules, which are always wanted by the user. Therefore, the analysis and developed solutions to this problem have deemed this objective to be fulfilled. This work has also contributed to the wider LCS community; providing a better understanding and verified solutions. Additionally, this answers one of the questions presented to this project in whether an LCS algorithm, in particular XCS, can be adapted to mine association rules. The answer to this has been found to be yes, as long as some changes are made to the algorithms fitness calculation and GA.

There were also problems found with the ability to mine online, which was cited in Orriols-Puig and Casillas (2010) as a major advantage. However, there has been evidence presented in this project that there are problems with fitness updating in XCS. Therefore, solutions were presented to further aid online learning; although there are substantially more evaluations to be carried out. These additional evaluation include; streaming data and the algorithms ability to provide output whilst mining this stream. This is believed to be a substantial project within itself, and therefore, was not attempted in this project.

The evaluation presented in Orriols-Puig and Casillas (2010) for continuous attributes, was highly criticised due to the determination of a rule output being satisfactory due to having high fitness. While it is accepted that this is high fitness paramount, an evaluation against other methods, such as QuantMiner and clustering, which determine whether the rules output are actually interesting and describe the correct regions, is needed to fully investigate and evaluate the algorithms ability. Therefore this evaluation was undertaken on both real and sythethic datasets of significant size, to fully evaluate the effectiveness of the algorithm to produce interesting associations within a dataset.

It was found to be easy to evaluate the output of categorical rules, as Apriori can be used as the gold standard, whereas the evaluation for continuous attributes was much more challenging. The novel introduction of comparing associations rules to clustering is seen as a contribution to this project, along with providing justification that association rules cannot only provide similar clusters, but also overcome some limitations of clustering, such as number of clusters and varying amounts of attributes. However, as previously discussed, due to the need for running a clustering algorithm of each combination of attributes being infeasible, the evaluation was limited to the top rules. It can be seen as future work to expand this evaluation and provide an automated process to carry out this evaluation, producing a more in-depth comparison between the two methods. In conclusion, this

third evaluation objective can be seen as fulfilled, however improvements could have been made to further investigate the links with clustering.

Overall this project can be deemed a success completing the objectives set out; however, the optional objective of learning temporal attributes was not completed, although due to the flexibility of the implementation this should be much easier to add. This project has contributed to past research both in terms of implementation, evaluation/investigation and theoretical aspects of both association rule miners and Learning Classifier Systems. Leading from this it has also set out new areas of investigation and implementation, which can be attempted in the future.

6.2 Contributions

This project has made substantial contributions to both association rule mining and Learning Classifier Systems, in terms of implementation contributions:

- Developing a expandable useable association rule miner, with limited need of parameter setting by user that can mine online, with a variety of attribute types.
- Expand and improved on the QuantMiner implementation giving the ability to describe multiple clusters wihtin a set of attributes
- Automatic setting of the maxInterval parameter improving upon the CSAR implementation, which needed to normalise a dataset, as well as having a maxInterval algorithm wide rather than each attribute as XCSAR.
- Improvements to provide solutions for the overlapping problem, giving more confidence that overlapping rule will be presented to the user.

Investigation contributions:

- Extensive evaluation on the rules produced by XCSAR, including the use of clustering both visually and using k-mean, which expands upon the evaluation presented in (Orriols-Puig and Casillas, 2010).
- Investigation into the overlapping issue, finding the main elements that cause the problem, contributing both to the ability of XCSAR and to the LCS community in general.
- Investigation into the ability to mine online, which was not presented in (Orriols-Puig and Casillas, 2010).

Theoretical contributions:

- Improvements to the ability to set the maxInterval automatically, using standard deviation.
- Improvements to the deletion probability, so if rules are created and do not become experience they're deletion probability is calculated inline with their lack of fitness.
- Improvements to the online learning, which need to be further investigated.
- Providing the mechanisms that cause the overlapping problem, including the theoritical reasons for this.

6.3 Future Work

Future work has been highlighted above, however below is a summary of future work that can further add to the gains of this project:

- Introducing the ability to mine temporal attributes, which due to the object oriented design of XCSAR should be relatively simple. However due to the limited alternate algorithms a methodology for evaluating the rules will have to be explored.
- Further scenarios need to be evaluated for the mining of quantitative association rules from sythentic datasets, to determine robustness of producing multiple clusters.
- The use of clustering to evaluate association rules has been introduced in this project, however comparing the output of clustering algorithms is also an combinatorial problem, therefore to provide a better comparison an automated process should be developed.
- The output of many association rules miners produces huge amounts of rules, therefore the ability to explore these rules using a graphical user interface would be advantageous and could provide extra usability to the user.
- This algorithm has presented the ability to programmatically set the maxInterval parameter, however it would also be of interest to remove this parameter and provide a fitness function that takes into consideration the attributes interval size.

Appendices

Appendix A

Example of Mining Process

This appendix will provide an example of the association rule mining process, it will follow the basic Apriori algorithm; finding frequent itemsets that are greater than a *minSupport* value, association rules will then be found from the frequent itemsets mined. To allow an informative but short example, *minSupport*, will be set to 0.6. Table A.1, provides the fictional dataset containing 5 transactions.

tid	Cereal	Bacon	Milk	Bread	Butter	Doughnuts
1	1	1	1	0	1	0
2	1	1	0	1	1	0
3	1	0	1	0	0	1
4	1	0	1	1	1	0
5	1	0	1	1	1	0

Table A.1: Transaction Database

The first step of the mining process is to find frequent itemsets, this is carried using a level wise search while implementing Apriori. Therefore Table A.2a, shows the individual items and whether they meet the minimum support. It can be seen from Table A.2b that the the search has been pruned, as the *AprioriProperty* says that any superset of an infrequent set is not frequent so those branches do not need to be searched.

Itemset	Support	$> minSupport$
$\{Cereal\}$	1	y
$\{Bacon\}$	0.4	n
$\{Milk\}$	0.8	y
$\{Bread\}$	0.6	y
$\{Butter\}$	0.8	y
$\{Doughnuts\}$	0.2	n

(a) Itemsets - Size 1

Itemset	Support	$> minSupport$
$\{Cereal, Milk\}$	0.8	y
$\{Cereal, Bread\}$	0.6	y
$\{Cereal, Butter\}$	0.8	y
$\{Milk, Butter\}$	0.6	y
$\{Bread, Butter\}$	0.6	y

(b) Itemsets - Size 2

Table A.2: Frequent Itemsets of Size 1 and 2

Table A.3, gives the itemsets of size 3 which are the largest frequent itemsets with *minSupport* greater than 0.6. The process now starts with the frequent itemsets of the smallest length and creates rules of the from the combination of items in the set. The confidence is then calculated to determine whether this rule can be classed as interesting, these rule and their respective confidence can be found in Table A.4. Table A.5 then shows the rules created from the itemsets of size 3. An interesting observation to make is that as discussed in Section 2.3.1, the confidence of all rules with *Cereal* in the consequent have confidence of one.

Itemset	Support	$> minSupport$
$\{Cereal, Milk, Bread\}$	0.4	n
$\{Cereal, Milk, Butter\}$	0.6	y
$\{Cereal, Bread, Butter\}$	0.6	y
$\{Milk, Bread, Butter\}$	0.4	n

Table A.3: Itemsets - Size 3

Rule	Confidence	$> minConfidence$
$\{Cereal\} \rightarrow \{Milk\}$	0.8	y
$\{Milk\} \rightarrow \{Cereal\}$	1	y
$\{Cereal\} \rightarrow \{Bread\}$	0.6	n
$\{Bread\} \rightarrow \{Cereal\}$	1	y
$\{Cereal\} \rightarrow \{Butter\}$	0.8	y
$\{Butter\} \rightarrow \{Cereal\}$	1	y
$\{Milk\} \rightarrow \{Butter\}$	0.75	y
$\{Butter\} \rightarrow \{Milk\}$	0.75	y
$\{Bread\} \rightarrow \{Butter\}$	1	y
$\{Butter\} \rightarrow \{Bread\}$	0.75	y

Table A.4: Association Rules with 2 items

Rule	Confidence	$> minConfidence$
$\{Cereal\} \rightarrow \{Milk, Butter\}$	0.6	n
$\{Milk\} \rightarrow \{Cereal, Butter\}$	0.75	y
$\{Butter\} \rightarrow \{Cereal, Milk\}$	0.75	y
$\{Cereal, Milk\} \rightarrow \{Butter\}$	0.75	y
$\{Cereal, Butter\} \rightarrow \{Milk\}$	0.75	y
$\{Milk, Butter\} \rightarrow \{Cereal\}$	1	y
$\{Cereal\} \rightarrow \{Bread, Butter\}$	0.6	n
$\{Bread\} \rightarrow \{Cereal, Butter\}$	1	y
$\{Butter\} \rightarrow \{Cereal, Bread\}$	0.75	y
$\{Cereal, Bread\} \rightarrow \}\{Butter\}$	1	y
$\{Cereal, Butter\} \rightarrow \{Bread\}$	0.75	y
$\{Bread, Butter\} \rightarrow \{Cereal\}$	1	y

Table A.5: Association Rules with 3 items

Appendix B

Online Learning Dataset

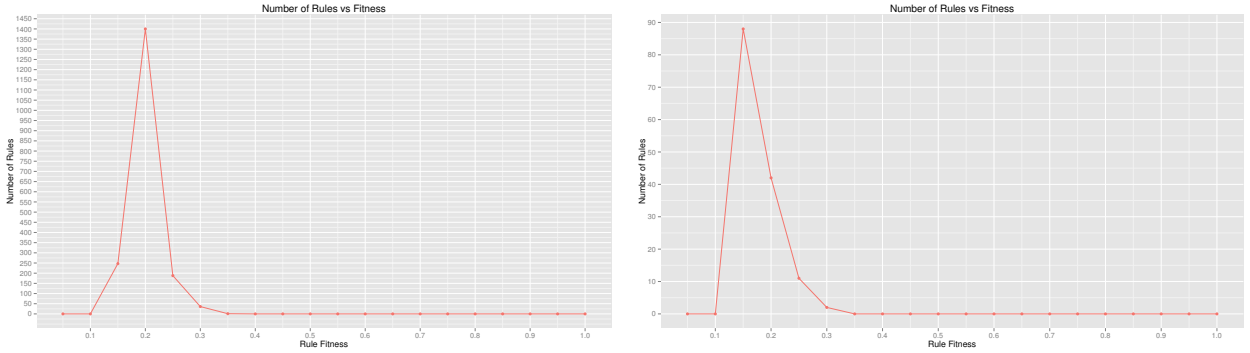
tid	Cereal	Bacon	Milk	Bread	Butter	Doughnuts
1	0	0	0	1	0	1
2	0	0	1	0	0	1
3	0	1	0	1	1	0
4	0	1	0	0	0	1
5	0	1	0	0	0	1

Table B.1: Transaction Database

Appendix C

Extra Results from Evaluation Chapter

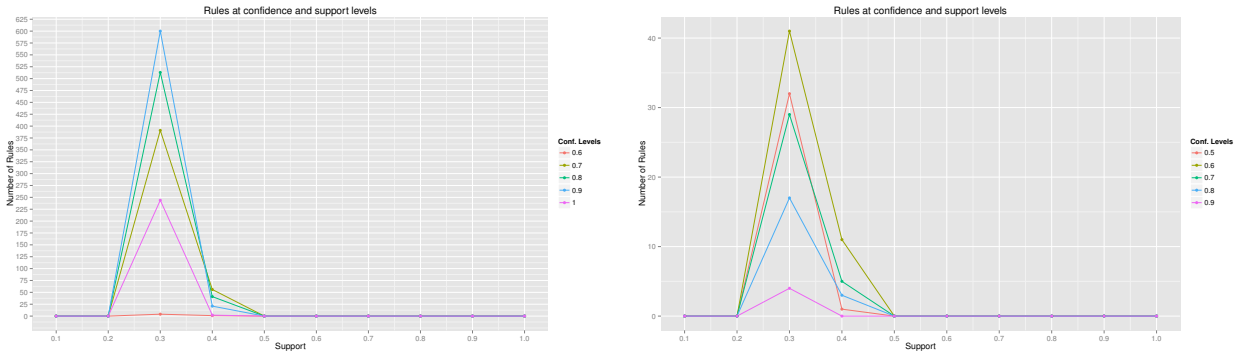
Wine Dataset Results showing that XCSAR concentrates on similar regions compared to QuantMiner, however due to Apriori feeding the GA in QuantMiner there are more rules due to this exhaustive approach.



(a) Number of Rules vs Fitness for the QuantMiner algorithm

(b) Number of Rules vs Fitness for the XCSAR algorithm

Figure C.1: Number of rules at fitness levels for the wine dataset for both the XCSAR and QuantMiner algorithm

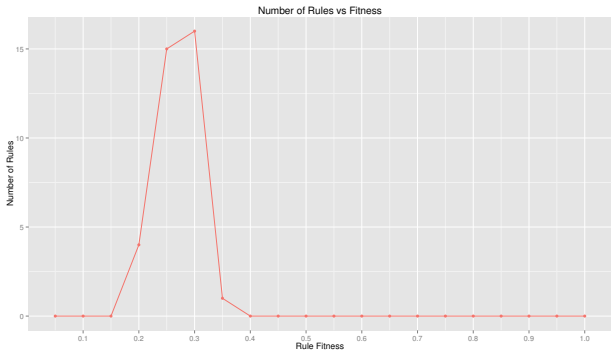


(a) Number of Rules at specified support and confidence levels for the QuantMiner algorithm

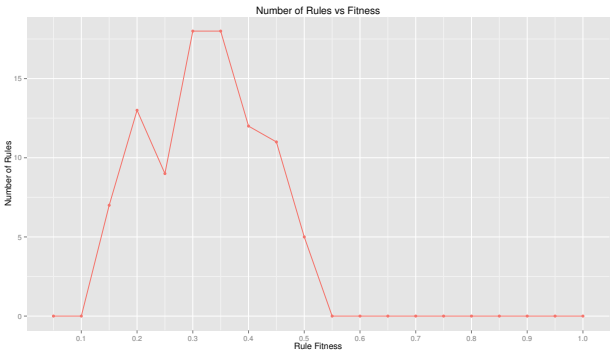
(b) Number of Rules at specified support and confidence levels for the XCSAR algorithm

Figure C.2: Number of rules at support and confidence levels, where confidence is specified by the colour of the line for the Wine dataset.

Iris Dataset Results from the iris dataset, showing favourable results compared to QuantMiner.

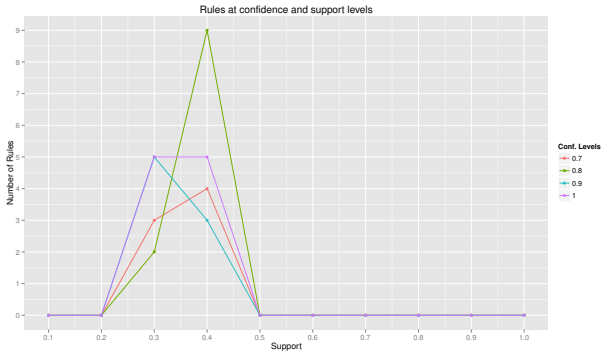


(a) Number of Rules vs Fitness for the QuantMiner algorithm

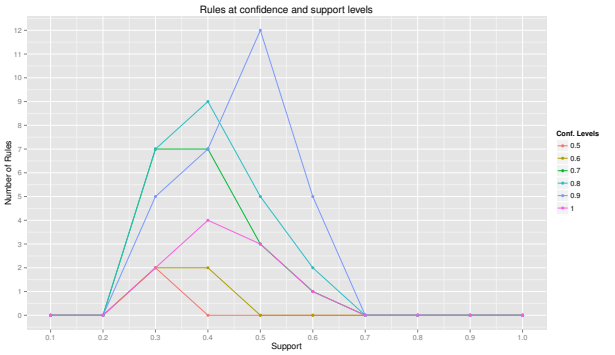


(b) Number of Rules vs Fitness for the XCSAR algorithm

Figure C.3: Number of rules at fitness levels for the Iris dataset for both the XCSAR and QuantMiner algorithm



(a) Number of Rules at specified support and confidence levels for the QuantMiner algorithm



(b) Number of Rules at specified support and confidence levels for the XCSAR algorithm

Figure C.4: Number of rules at support and confidence levels, where confidence is specified by the colour of the line for the Iris dataset.

Bibliography

Agrawal, R., Imielinski, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. In *IN: PROCEEDINGS OF THE 1993 ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, WASHINGTON DC (USA*, pages 207–216.

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules.

Aumann, Y. and Lindell, Y. (1999). A statistical theory for quantitative association rules. In *Journal of Intelligent Information Systems*, pages 261–270.

Bernad-Mansilla, E. and Garrell-Guiu, J. M. (2003). Accuracy-based learning classifier systems: models, analysis and applications to classification tasks. *Evol. Comput.*, 11(3):209–238.

Brin, S., Motwani, R., and Silverstein, C. (1997a). Beyond market baskets: generalizing association rules to correlations. *SIGMOD Rec.*, 26(2):265–276.

Brin, S., Motwani, R., Ullman, J. D., and Tsur, S. (1997b). Dynamic itemset counting and implication rules for market basket data. *SIGMOD Rec.*, 26(2):255–264.

Butz, M. and Wilson, S. (2001). An algorithmic description of xcs. In Luca Lanzi, P., Stolzmann, W., and Wilson, S., editors, *Advances in Learning Classifier Systems*, volume 1996 of *Lecture Notes in Computer Science*, pages 253–272. Springer Berlin Heidelberg.

Geng, L. and Hamilton, H. J. (2006). Interestingness measures for data mining: A survey. *ACM Comput. Surv.*, 38(3).

Hidber, C. (1999). Online association rule mining. In *Online Association Rule Mining*, pages 145–156.

Holland, J. (1976). *Adaption*. Plennum.

Holland, J. H. and Reitman, J. S. (1977). Cognitive systems based on adaptive algorithms. *SIGART Bull.*, (63):49–49.

IBM (2013). Ibm four vs of big data. `http://www.ibmbigdatahub.com/sites/default/files/styles/infographic-xlarge/public/infographic_image/4-Vs-of-big-data.jpg`.

Ioannides, C., Barrett, G., and Eder, K. (2011a). Improving xcs performance on overlapping binary problems. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1420–1427. IEEE.

Ioannides, C., Barrett, G., and Eder, K. (2011b). Xcs cannot learn all boolean functions. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, GECCO '11, pages 1283–1290, New York, NY, USA. ACM.

Kovacs, T. (2002). *Strength or Accuracy: Credit Assignment in Learning Classifier Systems*. PhD thesis, University Of Birmingham. Other identifier: 2001082.

Kovacs, T. and Tindale, R. (2013). Analysis of the niche genetic algorithm in learning classifier systems.

Lin, W., Alvarez, S. A., and Ruiz, C. (2000). Collaborative recommendation via adaptive association rule mining. In *Data Mining and Knowledge Discovery*.

Mata, J., Alvarez, J. L., and Riquelme, J. C. (2002). An evolutionary algorithm to discover numeric association rules. In *Proceedings of the 2002 ACM symposium on Applied computing*, SAC '02, pages 590–594, New York, NY, USA. ACM.

Orriols-Puig, A. and Casillas, J. (2010). Evolution of interesting association rules online with learning classifier systems. In Bacardit, J., Browne, W., Drugowitsch, J., Bernadó-Mansilla, E., and Butz, M., editors, *Learning Classifier Systems*, volume 6471 of *Lecture Notes in Computer Science*, pages 21–37. Springer Berlin Heidelberg.

Patrick O. Stalph, M. V. B. (2003). *Documentation of JavaXCSF*. COBOSLAB, University of Wurzburg.

Ryan J. Urbanowic, J. H. M. (2008). Learning classifier systems: A complete introduction, review, and roadmap. *Journal of Artificial Evolution and Applications*, 2009:25.

Salleb-Aouissi, A., Vrain, C., and Nortet, C. (2007). Quantminer: A genetic algorithm for mining quantitative association rules. In *IJCAI*, pages 1035–1040.

Srikant, R. and Agrawal, R. (1996). Mining quantitative association rules in large relational tables. *SIGMOD Rec.*, 25(2):1–12.

Tan, P.-N., Kumar, V., and Srivastava, J. (2004). Selecting the right objective measure for association analysis. *Inf. Syst.*, 29(4):293–313.

Tuzhilin, A. (1995). On subjective measures of interestingness in knowledge discovery. In *In Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pages 275–281. AAAI Press.

Wilson, S. (2000). Get real! xcs with continuous-valued inputs. In Lanzi, P., Stolzmann, W., and Wilson, S., editors, *Learning Classifier Systems*, volume 1813 of *Lecture Notes in Computer Science*, pages 209–219. Springer Berlin Heidelberg.

Wilson, S. W. (1995). Classifier fitness based on accuracy. *Evol. Comput.*, 3(2):149–175.